# Modeling and Computation in Science and Engineering

**Hermann Riecke**

Engineering Sciences and Applied Mathematics

h-riecke@northwestern.edu

Winter 2017

December 2, 2020

# Contents



[1] Runge (1895) and Kutta (1901)

[2] John Couch Adams (1819-1892), predicted the existence of Neptune based on perturbation calculations (independently also predicted by U. Le Verrier).

[3] Forest R. Moulton (1872-1952), astronomer.

# References

[1] Toni Feder. Statistical physics for the birds. *Physics Today*, October:28, 2007.

[2] E. Hairer and G. Wanner. Stiff differential equations solved by Radau methods. *Journal of Computational and Applied Mathematics*, 111(1-2):93–111, November 1999.

[3] H. Levine, W. J. Rappel, and I. Cohen. Self-organization in systems of self-propelled particles. *Phys. Rev. E*, 6302(2):017101, January 2001.

# Index

# 1 Introduction

In this class we discuss in depth numerical methods to solve ordinary differential equations and we will implement the methods in applications drawn from various disciplines. We will deal mostly with evolution equations, i.e. initial-value problems.

The main topics will be:

- Basic approximation methods
  - accuracy & error estimation, order of method
  - stability: avoid catastrophic accumulation of error
  - convergence to the correct solution
- Problems with multiple time scales: stiff equations
- PDEs as coupled sets of ODEs for each point in space
- Stochastic differential equations: noise
  Brownian motion: particle bombarded by water molecules
- Two-point boundary-value problems:
  beam on end supports

## 1.1 Applications

The applications we will encounter in the homework deal with

- Self-organization of complex systems: particle motion as a model for flocking and swarming of animals
- Neuroscience: model for ion currents that generate action potential of nerve cells
- Chemical reactions: oscillations far from thermodynamic equilibrium
- Stochastic differential equations
- Optimal control: controling a robot arm

## 1.2 Basic Methods

Consider differential equations of the type

$$\frac{d\mathbf{y}}{dt} = \mathbf{F}(t, \mathbf{y}) \qquad \mathbf{y}(t=0) = \mathbf{y}_0$$

In general $\mathbf{y}$ and $\mathbf{F}$ can be vectors of size $n$; then one has a system of $n$ coupled equations.

**Note:**

- All higher-order equations can be written as systems
- For simplicity we consider for now scalar equations

We will focus only on sufficiently smooth functions $\mathbf{F}(t, \mathbf{y})$, i.e. functions that are *Lipschitz continuous*. Thus, we require that for any $\mathbf{y}_1$ and $\mathbf{y}_2$ the function $\mathbf{F}(t, \mathbf{y})$ satisfy

$$\|F(t, \mathbf{y}_1) - F(t, \mathbf{y}_2)\| \leq L \|\mathbf{y}_1 - \mathbf{y}_2\|,$$

where $L$ is some constant that does not depend on $\mathbf{y}_{1,2}$.

**Example 1**: Any differentiable function is Lipschitz continuous.HRAdd Figures

Consider a scalar differentiable function $F(y)$ with

$$L = \max_y \left| \frac{\partial F(t, y)}{\partial y} \right|$$

Then we know from the the mean-value theorem

$$\begin{aligned} \frac{|F(t, y_1) - F(t, y_2)|}{|y_1 - y_2|} &= \left| \frac{\partial F(t, \hat{y})}{\partial y} \right| \qquad \text{for some } \hat{y} \text{ between } y_1 \text{and } y_2 \\ &\leq L. \end{aligned}$$

**Example 2**: A Lipschitz continuous function need not be differentiable.

Consider $F(t, y) = |y|$. Then

$$|F(t, y_1) - F(t, y_2)| = ||y_1| - |y_2|| = \begin{cases} |y_1 - y_2| & \text{for } y_1 y_2 \geq 0 \\ |y_1 + y_2| & \text{for } y_1 y_2 < 0 \end{cases}$$

For $y_1 y_2 < 0$ we have $|y_1 + y_2| \leq |y_1 - y_2|$ and therefore in both cases $L = 1$ serves as a Lipschitz constant.

**Theorem: Uniqueness of the Solution**

The differential equation

$$\frac{dy}{dt} = F(t, y), \qquad y(t_0) = y_0$$

has a unique solution in any domain in which $F$ is Lipschitz continuous in $y$.

**Numerical approximations:**

To solve the differential equation

$$\frac{dy}{dt} = F(t, y)$$

numerically we discretize the time $\Rightarrow$ for $y_n \equiv y(t_n)$ we need an algorithm

$$y_{n+1} = f(t_1, ..., t_n, y_1, ..., y_n)$$

For small time steps $\Delta t$ we can use a Taylor expansion

$$\begin{aligned} y_{n+1} &= y(t_n + \Delta t) = \\ &= y(t_n) + \Delta t \left.\frac{dy}{dt}\right|_{t_n} + \frac{1}{2}\Delta t^2 \left.\frac{d^2y}{dt^2}\right|_{t_n} + ... \end{aligned}$$

Of course, we cannot keep all terms in the expansion. As a first approach we truncate the expansion as

$$y_{n+1} = y_n + \Delta t\, F(t_n, y_n)$$

This is the forward Euler[4] scheme.

The truncation introduces an error, the *local truncation error* $\tau_l$. To obtain an estimate for this error we compare the approximation with the exact solution $\tilde{y}$

$$\begin{aligned} \tau_l &= \tilde{y}(t_{n+1}) - y_{n+1} = \tilde{y}_n + \Delta t \left.\frac{d\tilde{y}}{dt}\right|_{t_n} + \frac{1}{2}\Delta t^2 \left.\frac{d^2\tilde{y}}{dt^2}\right|_{t_n} + ... - \\ &\quad - (y_n + \Delta t\, F(t_n, y_n)) \end{aligned}$$

Assume that the solution was still exact at $t_n$. Then we can estimate the error arising in this single time step:

$$\tilde{y}_n = y_n \qquad \Rightarrow \qquad \left.\frac{d\tilde{y}}{dt}\right|_{t_n} = F(t_n, \tilde{y}_n) = F(t_n, y_n)$$

$$\tau_l = \frac{1}{2}\Delta t^2 \left.\frac{d^2\tilde{y}}{dt^2}\right|_{t_n} + ... = \mathcal{O}(\Delta t^2)$$

**Notes:**

- Definition of the order of $\Delta t^2$, i.e. $\mathcal{O}\left(\Delta t^2\right)$ :

  $$\tau = \mathcal{O}(\Delta t^2) \qquad \Leftrightarrow \qquad \tau = a\Delta t^2 + b\Delta t^3 + ...$$

  with $a$, $b$, ... some numbers that are independent of $\Delta t$.
  Thus, as $\Delta t \to 0$ the local error $\tau_l$ in this time step also goes to zero and it does so at least as fast as quadratically.

[4] Leonhard Euler (1707-1783)

- Thus, the local error of the forward Euler scheme $\mathcal{O}(\Delta t^2)$.

**Local vs Global Error:**
In each time step a local error $\mathcal{O}(\Delta t^2)$ is incurred $\Rightarrow$ for a fixed time interval $[0, T] = [0, N\Delta t]$ the algorithm takes $N$ steps, during each of which a local error is incurred. The error for the whole run is called the global error z

$$\tau_g = \sum_{n=1}^{N} \tau_{l,n} = \sum_{n=1}^{N} \mathcal{O}(\Delta t^2) = \mathcal{O}(N\Delta t^2) = \mathcal{O}(\Delta t)$$

**Note:**

- The *global error* $\tau_g$ is expected to be one order lower than the *local error* $\tau_l$.
- The forward Euler scheme is a $1^{st}$-order method.

Expect:

- For small $\Delta t$ the error is smaller for schemes with higher order
- The computational effort is larger for higher-order method
- Depending on the problem and the accuracy desired simple or more elaborate schemes are more efficient.

**Simple Example:**

$$\frac{dy}{dt} = -y \qquad \Rightarrow \qquad y(t) = y_0 e^{-t}$$

**Forward Euler**

$$y_{n+1} = y_n - \Delta t\, y_n = (1 - \Delta t)\, y_n$$

Graphically, the forward Euler scheme corresponds to extrapolating along the tangent of the solution (Fig.1).

**Notes:**

- For small $\Delta t$ numerical solution decays as it should and is close to the exact solution
- For large $\Delta t$ numerical solution can *oscillate* and *grow* although the exact solution decays monotonically: the method is **unstable**.

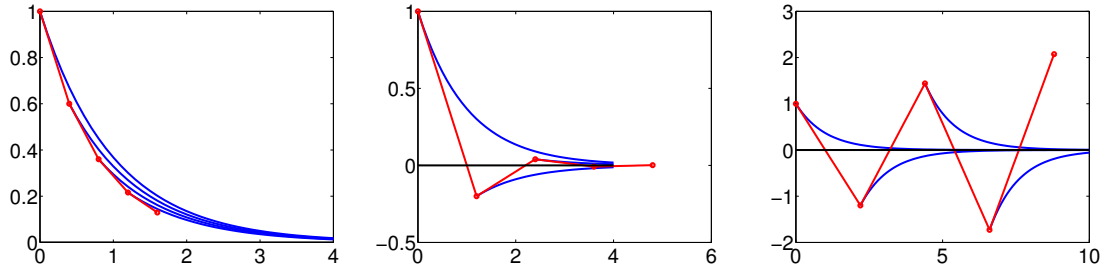


Figure 1: Sketch of forward Euler algorithm for $\dot{y} = -y$. For small time step ($\Delta t = 0.4$) one obtains a good approximation. For larger time steps the solution exhibits erroneous oscillations ($\Delta = 1.2$), which can grow and lead to blow-up ($\Delta t = 2.2$).

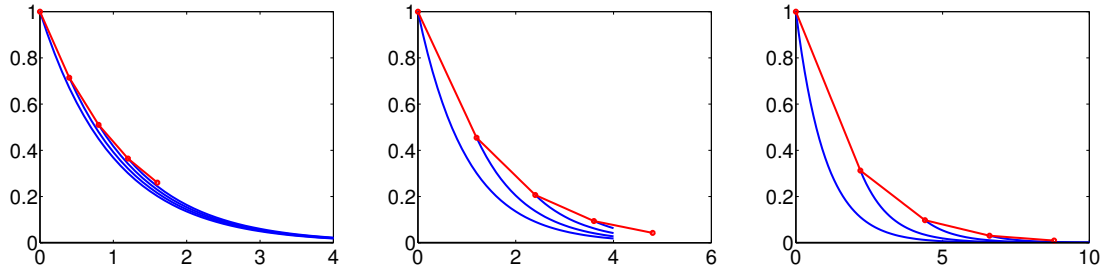


Figure 2: Sketch of backward Euler algorithm for $\dot{y} = -y$. For small time step ($\Delta t = 0.4$) one obtains a good approximation. For larger time steps the approximation becomes worse, but no oscillations and no blow-up arise, even for arbitrarily large time steps.

**Backward Euler:**

Now expand the solution about $t_{n+1}$ and $y_{n+1}$

$$y_n = y(t_{n+1} - \Delta t) = y_{n+1} - \Delta t \left.\frac{dy}{dt}\right|_{t_{n+1}} + \mathcal{O}(\Delta t^2)$$

This yields

$$y_{n+1} = y_n + \Delta t\, F(t_{n+1}, y_{n+1})$$

Graphically, the new value is obtained via the tangent corresponding to the next time step. This does not allow any overshoot (Fig.2).

**Notes:**

- The backward Euler scheme gives an *implicit* equation for $y_{n+1}$
- For linear differential equations can simply solve for $y_{n+1}$. E.g. $F(t, y) = -ay$,

$$y_{n+1} = y_n - \Delta t\, a y_{n+1} \qquad \Rightarrow \qquad y_{n+1} = \frac{1}{1 + \Delta t\, a}\, y_n$$

- This scheme does not oscillate or blow-up for arbitrarily large $\Delta t$: it is stable for any $\Delta t$ (Fig.2).

- For nonlinear differential equations the backward Euler scheme results in a nonlinear equation that has to be solved in each time step.
  Examples:

  1. $$\frac{dy}{dt} = -y^3$$
     $$y_{n+1} = y_n - \Delta t\, y_{n+1}^3$$
     This yields cubic equation for $y_{n+1}$ in each time step, which allows a analytical solution. But it is somewhat complicated
  2. $$\frac{dy}{dt} = \sin y$$
     $$y_{n+1} = y_n + \Delta t\, \sin y_{n+1}$$
     This yields a transcendental equation for $y_{n+1}$, which has to be solved numerically in each time step.

**Thus:**

- Implicit schemes are in general much harder to implement.
- Their advantage is that they tend to have much **greater stability**.

# 2 One-Step Methods

The two Euler schemes are only first-order accurate. How do we get more accurate methods?

- One-step methods:

  $$y_{n+1} = F(y_{n+1}, y_n) \qquad \text{independent of older values } y_{n-1}, y_{n-2}, ...$$

- Multi-step methods

  $$y_{n+1} = F(y_{n+1}, y_n, y_{n-1}, \ldots) \qquad \text{involves also older values } y_{n-1}, y_{n-2}, ...$$

## 2.1 Taylor-Series Methods

To establish a connetion between $y_{n+1}$ and $y_n$ we used a Taylor expansion about

- $t_n$ and $y_n$ : $\Rightarrow$ explicit method
- $t_{n+1}$ and $y_{n+1}$: $\Rightarrow$ implicit method

1. Forward Euler was obtained by $1^{st}$-order Taylor series

$$y_{n+1} = y_n + \Delta t \underbrace{F(t_n, y_n)}_{\frac{dy}{dt}\big|_{t_n}}$$

2. Use the $2^{nd}$-order Taylor series

$$y_{n+1} = y_n + \Delta t \left.\frac{dy}{dt}\right|_{t_n} + \frac{1}{2}\Delta t^2 \left.\frac{d^2y}{dt^2}\right|_{t_n} + \mathcal{O}(\Delta t^3)$$

To make use of this expansion we need

$$\begin{aligned} \frac{d^2y}{dt^2} &= \frac{d}{dt}\left(\frac{dy}{dt}\right) = \frac{d}{dt}\left(F(t,y)\right) \\ &= \frac{\partial F}{\partial t} + \frac{\partial F}{\partial y}\underbrace{\frac{dy}{dt}}_{F} \end{aligned}$$

Thus, we obtain

$$y_{n+1} = y_n + \Delta t\, F(t_n, y_n) + \frac{1}{2}\Delta t^2 \left(\frac{\partial F(t_n, y_n)}{\partial t} + \frac{\partial F(t_n, y_n)}{\partial y} F(t_n, y_n)\right) + \mathcal{O}(\Delta t^3)$$

**Note:**

- One can use this approach to generate also methods of yet higher order. But the evaluation becomes involved and higher and higher derivatives of $F$ are needed. For systems many different partial derivatives would be needed.

**Truncation Error:**

$$\begin{aligned} \tau &= \tilde{y}(t_{n+1}) - y_{n+1} = \tilde{y}(t_n) + \Delta t\frac{d\tilde{y}}{dt} + \frac{1}{2}\Delta t^2\frac{d^2\tilde{y}}{dt^2} + \frac{1}{6}\Delta t^3\frac{d^3\tilde{y}}{dt^3} - \\ &\left[\underbrace{y_n}_{\tilde{y}_n} + \Delta t\underbrace{F}_{\frac{dy}{dt}} + \frac{1}{2}\Delta t^2\underbrace{\left(\frac{\partial F}{\partial t} + \frac{\partial F}{\partial y}F\right)}_{\frac{d^2y}{dt^2}}\right] \end{aligned}$$

$$\tau = \frac{1}{6}\Delta t^3\frac{d^3\tilde{y}}{dt^3} + h.o.t. = \mathcal{O}(\Delta t^3)$$

3. Expand to cubic order about the new time step: implicit

$$\begin{aligned} y(t_n) &= y(t_{n+1} - \Delta t) = \\ &= y_{n+1} - \Delta t \left.\frac{dy}{dt}\right|_{t_{n+1}} - \frac{1}{2}\Delta t^2 \left.\frac{d^2y}{dt^2}\right|_{t_{n+1}} + \mathcal{O}(\Delta t^3) \end{aligned}$$

thus

$$y_{n+1} = y_n + \Delta t F(t_{n+1}, y_{n+1}) + \frac{1}{2}\Delta t^2 \left[\frac{\partial F}{\partial t} + \frac{\partial F}{\partial y} F\right]\Bigg|_{t_{n+1}} + \mathcal{O}(\Delta t^3)$$

**Note:**

- If we keep only the first non-trivial term $\Delta t\, F$ we obtain the backward Euler scheme.
- As for the backward Euler method, the implicit equation becomes in general *nonlinear*.
- The Taylor series methods can only be used if the corresponding derivatives of $F$ exist. For equations like

  $$\frac{dy}{dt} = (t - t_0)^{1/2} y \qquad y(t_0) = 1$$

  one cannot generate the $2^{nd}$-order Taylor series since $\frac{\partial F}{dt}$ does not exist at $t = t_0$. Or $F$ could be discontinuous in $t$.

## 2.2 Quadrature Methods

Since the direct Taylor-series method does not yield efficient practicable high-order schemes we rewrite the ODE as an integral equation to get new ideas for the approximation of the ODE.

Thus

$$\frac{dy}{dt} = F(t, y)$$

leads to

$$y(t_{n+1}) = y(t_n) + \int_{t_n}^{t_{n+1}} F(t', y(t'))\, dt'$$

**Now:** use approximation methods for integrals

1. **End-point Rules**

   $$\int_{t_n}^{t_{n+1}} F(t', y(t'))\, dt' = \Delta t\, F(t_n, y(t_n)) + \mathcal{O}(\Delta t^2)$$

   The left-end-point rule yields the forward Euler method

   $$y_{n+1} = y_n + \Delta t\, F(t_n, y_n) + \mathcal{O}(\Delta t^2)$$

   Correspondingly, the right-end-point rule generates the backward Euler method.

2. **Trapezoidal Rule**

$$\int_{t_n}^{t_{n+1}} F(t', y(t'))\, dt' = \frac{1}{2}\Delta \left( F(t_n, y(t_n)) + F(t_{n+1}, y(t_{n+1})) \right) + \mathcal{O}(\Delta t^3)$$

$$y_{n+1} = y_n + \frac{1}{2}\Delta t \left( F(t_n, y_n) + \underbrace{F(t_{n+1}, y_{n+1})}_{\text{implicit term}} \right) + \mathcal{O}(\Delta t^3)$$

Can we avoid the implicitness?
Approximate $y_{n+1}$ on r.h.s.: since the implicit term is already multiplied by a factor of $\Delta t$, the forward Euler approximation is sufficient to approximate $y_{n+1}$ : the replacement

$$y_{n+1} \to y^*_{n+1} = y_n + \Delta t F(t_n, y_n) + \mathcal{O}(\Delta t^2)$$

introduces an error that contributes only to the already omitted term $\mathcal{O}(\Delta t^3)$

$$y_{n+1} = y_n + \frac{1}{2}\left( F(t_n, y_n) + F(t_{n+1}, y_n + \Delta t\, F(t_n, y_n)) \right)$$

This can be written more streamlined as

$$\begin{aligned} k_1 &= F(t_n, y_n) \\ k_2 &= F(t_n + \Delta t, y_n + \Delta t\, k_1) \\ y_{n+1} &= y_n + \frac{1}{2}\Delta t\, (k_1 + k_2) \end{aligned} \tag{1}$$

- simpler error estimate
- 4th-order scheme: Runge-Kutta
- HW1 code template
- General Runge-Kutta formulation

**Note:**

- This is method is called the improved Euler method, Heun's method[5], or Runge-Kutta method of $2^{nd}$-order

**Truncation error:**
A simpler method to *estimate* the order of a scheme is to consider the specific simple ODE

$$\frac{dy}{dt} = \lambda y$$

which has the exact solution

$$y(t_{n+1}) = y(t_n)\, e^{\lambda \Delta t}$$

[5]Karl Heun, 1859-1929

The improved Euler method yields for this equation

$$\begin{aligned} y_{n+1} &= y_n + \frac{1}{2}\Delta t\left[\lambda y_n + \lambda\left(y_n + \Delta t\lambda y_n\right)\right] = \\ &= y_n + \Delta t\lambda y_n + \frac{1}{2}\Delta t^2\lambda^2 y_n = \\ &= \left(1 + \Delta t\lambda + \frac{1}{2}\Delta t^2\lambda^2\right) y_n = \underbrace{e^{\Delta t\lambda} y_n}_{\text{exact solution}} + \mathcal{O}(\Delta t^3) \end{aligned}$$

Thus, as expected, the scheme is $2^{nd}$-order accurate.
**Note:**

- This method to determine the order gives only an *upper* limit for the order of the scheme. In principle, for this specific simple differential equation error cancellations could occur that raise the order but that do not occur for general equations. If that is the case the method would be of lower order for a general ODE.
- This method does not probe the implementation of any *explicit* time-dependence of $F$.

3. **Simpson's Rule**

$$\int_{t_n}^{t_{n+1}} F(t', y(t'))\, dt' = \frac{1}{6}\Delta t\left[F(t_n, y_n) + 4F(t_{n+\frac{1}{2}}, y(t_{n+\frac{1}{2}})) + F(t_{n+1}, y_{n+1})\right] + \mathcal{O}(\Delta t^5)$$

How to obtain now $y(t_{n+\frac{1}{2}})$ and $y(t_{n+1})$?
Analogously, to the treatment of the improved Euler method we can try

$$\begin{aligned} y^*_{n+\frac{1}{2}} &= y_n + \frac{1}{2}\Delta t\, F(t_n, y_n) \\ y^*_{n+1} &= y_n + \Delta t\, F(t_{n+\frac{1}{2}}, y^*_{n+\frac{1}{2}}) \\ y_{n+1} &= y_n + \frac{1}{6}\Delta t\,\left[F(t_n, y_n) + 4F(t_{n+\frac{1}{2}}, y^*_{n+\frac{1}{2}}) + F(t_{n+1}, y^*_{n+1})\right] \end{aligned}$$

Estimate the error of this scheme using the simplified method

$$\begin{aligned} y_{n+1} &= y_n + \frac{1}{6}\Delta t\left[\lambda y_n + 4\lambda y^*_{n+\frac{1}{2}} + \lambda y^*_{n+1}\right] \\ &= y_n + \frac{1}{6}\Delta t\left[\lambda y_n + 4\lambda(y_n + \frac{1}{2}\Delta t\lambda y_n) + \lambda\left(y_n + \Delta t\lambda(y_n + \frac{1}{2}\Delta t\lambda y_n\right)\right] \\ &= y_n\left[1 + \frac{1}{6}\Delta t\left(\lambda + 4\lambda + 2\Delta t\lambda^2 + \lambda + \Delta t\lambda^2 + \frac{1}{2}\Delta t^2\lambda^3\right)\right] \\ &= y_n\left[1 + \Delta t\lambda + \frac{1}{2}\Delta t^2\lambda^2 + \frac{1}{12}\Delta t^3\lambda^3\right] \\ &= y_n\left[e^{\Delta t\lambda} + \Delta t^3\lambda^3\left(\frac{1}{12} - \frac{1}{6}\right)\right] \end{aligned}$$

Thus, although Simpson's rule has an error of $\mathcal{O}(\Delta t^5)$ this treatment gives a much bigger truncation error

$$\tau = \mathcal{O}(\Delta t^3)$$

We need to get better approximations for $y_{n+\frac{1}{2}}$ and $y_{n+1}$; the approximations $y^*_{n+\frac{1}{2}}$ and $y^*_{n+1}$ had already an error of $\mathcal{O}(\Delta t^2)$. We will obtain these better approximations in a different way.

## 2.3 Runge-Kutta[6] Methods

As in the quadrature approach we evaluate $F$ at *intermediate points* between $t_n$ and $t_{n+1}$ and between $y_n$ and $y_{n+1}$. The question is which intermediate points will yield optimal accuracy.

We consider the general class of methods that can be formulated as

$$y_{n+1} = y_n + \Delta t \sum_{k=0}^{s} \gamma_k F_k$$

with

$$\begin{aligned} F_0 &= F(t_n, y_n) \\ F_k &= F\left(t_n + \alpha_k \Delta t, y_n + \Delta t \sum_{m=0}^{k} \beta_{km} F_m\right) \qquad k = 1, ..., s \end{aligned}$$

**Notes:**

- These Runge-Kutta methods have $s + 1$ stages
- The coefficients $\alpha_k$, $\beta_{km}$, and $\gamma_k$ are to be chosen to minimize the error, i.e. to obtain the highest order of method possible.

Which coefficients are required in each stage can be summarized in the **Butcher Array** (1963)**:**

| | | | | | |
|---|---|---|---|---|---|
| $\alpha_0 = 0$ | $\beta_{00} = 0$ | | | | |
| $\alpha_1$ | $\beta_{10}$ | $\beta_{11}$ | | | |
| $\alpha_2$ | $\beta_{20}$ | $\beta_{21}$ | $\beta_{22}$ | | |
| ... | ... | ... | ... | | |
| $\alpha_s$ | $\beta_{s0}$ | ... | ... | $\beta_{s,s-1}$ | $\beta_{s,s}$ |
| | $\gamma_0$ | $\gamma_1$ | ... | $\gamma_{s-1}$ | $\gamma_s$ |

**Notes:**

[6] Runge (1895) and Kutta (1901)

- If the diagonal elements are non-zero, $\beta_{ii} \neq 0$, the method is implicit, otherwise it is explicit.

**Examples:**

1. $s = 0$
   we need to choose $\gamma_0$
   $$y_{n+1} = y_n + \Delta t \cdot \gamma_0 \cdot F_0$$
   this will generate the forward Euler method: we need to choose $\gamma_0 = 1$.

2. $s = 1$, explicit
   now we need $\alpha_1$, $\beta_{10}$, $\gamma_0$, $\gamma_1$

   $$\begin{aligned} F_1 &= F\left(t_n + \alpha_1 \Delta t, y_n + \Delta t \beta_{10} F_0\right) \\ y_{n+1} &= y_n + \Delta t \left[\gamma_0 F_0 + \gamma_1 F_1\right] \end{aligned}$$

   Choose the coefficients to minimize the truncation error:

   $$\begin{aligned} \tilde{y}(t_{n+1}) - y_{n+1} &= \tilde{y}(t_n) + \Delta t \frac{d\tilde{y}}{dt} + \frac{1}{2}\Delta t^2 \frac{d^2\tilde{y}}{dt^2} + \frac{1}{6}\Delta t^3 \frac{d^3\tilde{y}}{dt^3} + \mathcal{O}(\Delta t^4) - \\ &\quad - y_n - \Delta t \left\{\gamma_0 F(t_n, y_n) + \gamma_1 F\left(t_n + \alpha_1 \Delta t, y_n + \Delta t \beta_{10} F(t_n, y_n)\right)\right\} = \\ &= \Delta t\, F + \frac{1}{2}\Delta t^2 \left(\frac{\partial F}{\partial t} + \frac{\partial F}{\partial y} F\right) + \\ &\quad + \frac{1}{6}\Delta t^3 \left[\frac{\partial^2 F}{\partial t^2} + \frac{\partial^2 F}{\partial t \partial y} F + \frac{\partial^2 F}{\partial t \partial y} F + \frac{\partial^2 F}{\partial y^2} F^2 + \frac{\partial F}{\partial y}\left(\frac{\partial F}{\partial t} + \frac{\partial F}{\partial y} F\right)\right] - \\ &\quad - \Delta t \left(\gamma_0 F + \gamma_1 \left\{F + \alpha_1 \Delta t \frac{\partial F}{\partial t} + \Delta t \beta_{10} F \frac{\partial F}{\partial y} + \frac{1}{2}\alpha_1^2 \Delta t^2 \frac{\partial^2 F}{\partial t^2} + \right.\right. \\ &\quad \left.\left. + \alpha_1 \Delta t^2 \beta_{10} F \frac{\partial^2 F}{\partial t \partial y} + \frac{1}{2}\Delta t^2 \beta_{10}^2 F^2 \frac{\partial^2 F}{\partial y^2} + \mathcal{O}(\Delta t^3)\right\}\right) \end{aligned}$$

   Compare the coefficients at each order in $\Delta t$:

   - $\Delta t$ :
     $$F - \gamma_0 F - \gamma_1 F = 0 \qquad \Rightarrow \qquad \gamma_0 + \gamma_1 = 1$$
   - $\Delta t^2$ :
     $$\frac{1}{2}\left(\frac{\partial F}{\partial t} + \frac{\partial F}{\partial y} F\right) - \gamma_1 \alpha_1 \frac{\partial F}{\partial t} - \gamma_1 \beta_{10} F \frac{\partial F}{\partial y} = 0$$
     $$\rightarrow \qquad \frac{1}{2} - \gamma_1 \alpha_1 = 0 \qquad \frac{1}{2} - \gamma_1 \beta_{10} = 0$$
   - $\Delta t^3$ : the term $\frac{\partial F}{\partial y}\left(\frac{\partial F}{\partial t} + \frac{\partial F}{\partial y} F\right)$ cannot be absorbed by any choice of coefficient

There are 3 other different terms to be balanced.
3 equations for 4 coefficients $\rightarrow$ 1 coefficient free:

$$\gamma_1 = \theta, \qquad \gamma_0 = 1 - \theta, \qquad \alpha_1 = \frac{1}{2\gamma_1} = \frac{1}{2\theta}, \qquad \beta_{10} = \frac{1}{2\theta}$$

This yields the Butcher array:

| | | |
|---|---|---|
| $0$ | $0$ | |
| $\frac{1}{2\theta}$ | $\frac{1}{2\theta}$ | $0$ |
| | $1-\theta$ | $\theta$ |

**Notes:**

- For all $\theta$ the error is $\mathcal{O}(\Delta t^3)$: $2^{nd}$-order accurate
- We could try to choose $\theta$ to minimize the prefactor of the truncation error. But the error depends on the function $F$: there is no choice of $\theta$ that would minimize the error *in general*.
- For $\theta = \frac{1}{2}$ one gets the second-order Runge-Kutta method from above.

General properties of the coefficients:

(a)

$$\sum_{k=0}^{s} \gamma_k = 1$$

We have

$$y_{n+1} = y_n + \Delta t \underbrace{\sum_{k=0}^{s} \gamma_k F_k}_{\text{weighted 'slope' } \frac{dy}{dt}} \tag{2}$$

Since the weights cannot depend on $F$ consider the special case $F = F_0 = const$.

$$\frac{dy}{dt} = F_0 \qquad \Rightarrow \qquad y = y_0 + F_0 t$$

and the exact discrete solution is

$$y_{n+1} = y_n + F_0 \, \Delta t.$$

Therefore the weights $\gamma_k$ need to add up to 1 in order to give 1 full time step $\Delta t$.

(b)

$$\alpha_k = \sum_{m=0}^{k} \beta_{km}$$

Again, for $F = F_0$ the exact discrete solution at the intermediate time $t_n + \alpha_k \Delta t$ should be reproduced by the scheme,

$$y_n + \alpha_k \Delta t \, F_0 \overset{!}{=} y_n + \Delta t \sum_m \beta_{km} \, F_0 \tag{3}$$

Thus, the $\beta_{km}$ have to add up to $\alpha_k$.

In other words: the coefficients $\gamma_k$ and $\beta_{km}$ generate an effective slope and with it an effective 'rise'. This rise needs to match the effective 'run', which is $\Delta t$ in (2) and $\alpha_k \Delta t$ in (3).

3. $4^{th}$-order Runge-Kutta Method
As for the $2^{nd}$-order RK there are many possible RK4 with the same order of accuracy but different prefactors of the truncation error. The most commonly used, 'classical' method has the Butcher array

| | | | | |
|---|---|---|---|---|
| $0$ | $0$ | | | |
| $\frac{1}{2}$ | $\frac{1}{2}$ | $0$ | | |
| $\frac{1}{2}$ | $0$ | $\frac{1}{2}$ | $0$ | |
| $1$ | $0$ | $0$ | $1$ | $0$ |
| | $\frac{1}{6}$ | $\frac{1}{3}$ | $\frac{1}{3}$ | $\frac{1}{6}$ |

As expected, it satisfies the rules discussed for RK2:

$$\sum_k \gamma_k = 1 \qquad \sum_{m=0}^{k;} \beta_{km} = \begin{cases} \frac{1}{2} & \text{for } k = 1 \\ \frac{1}{2} & \text{for } k = 2 \\ 1 & \text{for } k = 3 \end{cases}$$

The method is typically written as

$$\begin{aligned} k_1 &= F(t_n, y_n) \\ k_2 &= F(t_n + \frac{1}{2}\Delta t, y_n + \frac{1}{2}\Delta t \, k_1) \\ k_3 &= F(t_n + \frac{1}{2}\Delta t, y_n + \frac{1}{2}\Delta t \, k_2) \\ k_4 &= F(t_n + \Delta t, y_n + \Delta t \, k_3) \\ y_{n+1} &= y_n + \frac{1}{6}\Delta t \, (k_1 + 2k_2 + 2k_3 + k_4) + \mathcal{O}(\Delta t^5) \end{aligned}$$

**Notes:**

- RK4 requires 4 evaluations of the r.h.s. $F(y, t, )$ compared to a single evaluation in the forward Euler scheme. Therefore, to be more efficient than the forward Euler scheme the higher-order accuracy of RK4 has to be significant. This will be the case if the time steps are sufficiently small. Thus, for large time steps the forward Euler scheme may be more efficient, whereas for small time steps RK4 will be more efficient.

- For $F = F(t)$ one recovers Simpson's rule: $k_2 = k_3$ yielding the typical factor of 4 of the Simpson rule for the contribution at $t_n + \Delta t/2$.
- Important: for systems $\frac{d\mathbf{y}}{dt} = \mathbf{F}(\mathbf{y})$ one needs to calculate first *all* components of $\mathbf{k}_i$ before one calculates $\mathbf{k}_{i+1}$.

**Notes:**

- It is possible to determine the maximal order that can be achieved for a given number of stages

| # of stages | 1 | 2 | 3 | 4 | 5 | 6 |
|---|---|---|---|---|---|---|
| maximal order | 1 | 2 | 3 | 4 | 4 | 5 |

Since the order does not increase when increasing the number of stages from 4 to 5, using 5 or 6 stages is less efficient than RK4.

# 3 Self-Organization of Swarms of Self-Propelled Particles ('Boids')

Consider a flock of birds, a school of fish, or a herd of wildebeest[7]. How does such a group of 100 to 10,000 animals stay together and fly/swim in a more or less coherent formation? Is there a leader or how do the animals know in which direction they should fly? Do they require very refined feed-back control to form coherent formations? In many cases there is no leader and the coherent motion is an *emergent property* of the swarm that arises merely from the interaction between the animals. Such motion is also of interest to control swarms of robotic vehicles without a central control computer [8].

Let us consider a very simple model for point objects that are moving with essentially fixed speed in two dimensions and whose direction of motion is determined by the motion of the objects in their vicinity. Such computer models for swarms of objects were first developed by Craig Reynolds (1986, cf his web site HTTP://WWW.RED3D.COM/CWR/BOIDS/) in the context of computer animations (e.g. in *Batman Returns* or *The Lion King*). They called these animals 'boids'.[9]

Newton's law of motion

$$m\frac{d^2\mathbf{r}}{dt^2} = \mathbf{F}.$$

It is in general convenient to rewrite higher-order equations of motion in terms of

[7]There are a number of movies showing huge flocks, e.g. http://www.youtube.com/watch?v=XH-groCeKbE http://www.youtube.com/watch?v=8vhE8ScWe7w http://www.youtube.com/watch?v=Sk_Blp6w528.

[8]See e.g. http://phys.org/news/2012-02-airborne-robot-swarms-complex-video.html

[9]Couzin web site on collective dynamics: http://collectivebehaviour.com/modeling-behaviour/

$1^{st}$-order differential equations

$$\frac{d\mathbf{r}}{dt} = \mathbf{v} \tag{4}$$

$$\frac{d\mathbf{v}}{dt} = \mathbf{f}(\mathbf{r}), \tag{5}$$

where we assume all quantities have been made dimensionless in a suitable fashion. The boids are watching their neighbors for clues in which direction to fly. This results in an interaction between the boids that mostly changes the direction but not so much the speed of their motion. We can model this by a force $\mathbf{f}_i$ acting on boid $i$

$$\frac{1}{m}\mathbf{F}_i = \frac{\bar{\mathbf{v}}_i}{|\bar{\mathbf{v}}_i|} - \mathbf{v}_i \tag{6}$$

where $\bar{\mathbf{v}}_i$ is the mean velocity observed by boid $i$. For fixed $\bar{\mathbf{v}}_i$ (5) can be solved exactly. Consider the $x$-component of velocity

$$\frac{dv_{xi}}{dt} = \bar{v}_{xin} - v_{xi}.$$

Inhomogeneous, linear differential equation of first order. First find solution to homogeneous equation

$$\frac{dv_{xi}}{dt} = -v_{xi} \qquad v_{xi} = Ae^{-t}.$$

Need still a particular solution of the inhomogeneous equation. Here that is easy:

$$v_{xi}^h = \bar{v}_{xin}.$$

The general solution is therefore

$$v_{xi}(t) = Ae^{-t} + \bar{v}_{xin}.$$

The coefficient $A$ is determined by initial conditions, e.g. for $v_{xi}(t=0) = 0$ we get

$$0 = A + \bar{v}_{xin} \qquad A = -\bar{v}_{xin}$$

with

$$v_{xi}(t) = \bar{v}_{xin}\left(1 - e^{-t}\right).$$

Similarly for $v_{yi}(t)$,which results in

$$\mathbf{v}_i(t) = \frac{\bar{\mathbf{v}}_i}{|\bar{\mathbf{v}}_i|}\left(1 - e^{-t}\right).$$

Thus, for fixed $\frac{\bar{\mathbf{v}}_i}{|\bar{\mathbf{v}}_i|}$ the boid eventually will fly in the direction of the average direction given by all the other boids. The speed (magnitude of the velocity) is then fixed, $|\mathbf{v}_i| = 1$.

We still need to specify: 'boids watch their neighbors'. Let us assume that close-by neighbors can be observed more precisely and boid $i$ will pay more attention

to them than to neighbors far away: for the average flight direction we therefore weigh close-by neighbors more,

$$\bar{\mathbf{v}}_i = \frac{1}{\mathcal{N}} \sum_{j=1}^{N} e^{-\Delta r_{ij}^2/\Delta r_{ave}^2} \mathbf{v}_j \qquad \text{with} \qquad \mathcal{N} = \sum_{j=1}^{N} e^{-\Delta r_{ij}^2/\Delta r_{ave}^2} \tag{7}$$

Here $\Delta r_{ij}$ is the distance between boids $i$ and $j$ and$\Delta r_{ave}$ characterizes the distance beyond which a boid does not pay much attention to the other boids any more. Because of this weighting the mean velocity $\bar{\mathbf{v}}_i$ is the mean velocity of the flock as observed by boid $i$ and differs from boid to boid.[10]

Even with this interaction between boids it is easy to see that a swarm with all boids flying in the same direction is a solution to the equations, because then $\frac{\bar{\mathbf{v}}_i}{|\bar{\mathbf{v}}_i|}$ is again constant and independent of $i$.

**But:** in general the boids move in different directions and $\frac{\bar{\mathbf{v}}_i}{|\bar{\mathbf{v}}_i|}$ depends on $i$ and through changes in the distances between boids and through changes in their flight direction $\frac{\bar{\mathbf{v}}_i}{|\bar{\mathbf{v}}_i|}$ is not constant in time. Then the equations cannot be solved exactly any more and we need an approximate, numerical method.

To assess whether the model boids start flying coherently as a flock we need a quantitative measure, an *order parameter,* for their *orientational order.* Let us consider the global mean velocity $\mathbf{V}$,

$$\mathbf{V}(t) = \frac{1}{N} \sum_{i=1}^{N} \mathbf{v}_i(t).$$

If the boids fly in randomly distributed directions the various $\mathbf{v}_i$ will cancel each other and the magnitude of $\mathbf{V}$ will be small, whereas for perfectly coherent motion $|\mathbf{V}| = 1$. Therefore $|\mathbf{V}(t)|$ is a useful order parameter.

In the simulations you will find that with this simple interaction the boids will not only eventually fly all in the same direction but will also form clusters, more condensed flocks. To avoid that they are hitting each other it is sometimes useful to introduce an additional *repulsive* short-range interaction between them. In addition, over larger distances the animals may also have the tendency to fly towards each other independent of their relative flight direction. This can be modeled by an attractive interaction. All interactions combined we then obtain [3]

$$\mathbf{f_i} = \frac{\bar{\mathbf{v}}_\mathbf{i}}{|\bar{\mathbf{v}}_\mathbf{i}|} - \mathbf{v_i} - \sum_{\mathbf{j=1}}^{\mathbf{N}} (\mathbf{r_i} - \mathbf{r_j}) \left( \alpha \mathbf{e}^{-\mathbf{\Delta r_{ij}^2/\Delta_{att}^2}} - \beta \frac{\mathbf{e}^{-\mathbf{\Delta r_{ij}^2/\Delta_{rep}^2}}}{\mathbf{\Delta r_{ij}^2} + \mathbf{\delta_{rep}^2}} \right).$$

Here $\Delta r_{ij}$ is the shortest distance between boid $i$ and boid $j$ for periodic boundary conditions. Similarly $(\mathbf{r}_i - \mathbf{r}_j)$ is the vector corresponding to the shortest distance

[10] Apparently, real birds relie more on a fixed number of neighboring birds than on all birds within a certain distance[1].

for periodic boundary conditions. The term $\delta_{rep}^2$ provides a regularization for the singularity of the repulsive term at $\Delta r_{ij} = 0$.

It turns out that with this extension swarms can also form well-organized rotating vortices.

In reality, the boids are exposed to a fluctuating environment and also cannot measure the flight direction of their neighbors perfectly. It is therefore important to consider the effect of small random perturbations on such swarming models. Particularly relevant are perturbations of the direction of flight of each boid. In principle, this leads to stochastic differential equations, which we will treat much later. But to get an idea of the influence of noise on the swarm we can add after each time step a small random perturbation to the velocities and then normalize the velocities again to $|\mathbf{v}_i| = 1$,

$$\mathbf{v}_i \to \mathbf{v}_i + \eta\,(\zeta_{xi}, \zeta_{yi}) \to \frac{\mathbf{v}_i}{|\mathbf{v}_i|} \tag{8}$$

with $\zeta_{xi}$ and $\zeta_{yi}$ being independent random variables that are uniformly distributed in $[-0.5, 0.5]$. The noise strength is given by $\eta$. One finds that for small noise levels the coherence of the swarms persist, with somewhat reduced order parameter $|\mathbf{V}|$. At a finite noise level $\eta_c$ the order parameter goes to $0$ and for $\eta > \eta_c$ swarms are not coherent any more (Vicsek et al., 1995, paper available on class web site).

Worth noting: in equlibrium statistical physics there is a well-studied class of models for ferromagnets in which the microscopic magnetic elements ('spins') that make up the macroscopic magnet are free to rotate in a plane ($xy$-model), similar to the flight direction of the boids. These spins have the tendency to lign up in parallel. However, thermal motion of the atoms constantly perturb such a regular arrangement in a random way somewhat similar to (8). In three-dimensional arrays such spins still lign up predominantly to generate some non-zero macroscopic magnetization as long as the noise (the temperature) is not too large ($\eta < \eta_c$). Above the critical temperature $\eta_c$ the macroscopic magnetization vanishes. In two dimensions, however, any infinitesimally small amount of noise destroys the macroscopic 'order' (magnetization), i.e. $\eta_c = 0$ (Mermin-Wagner theorem). The reason for this sensitivity is that the spins only interact with their immediate neighbors and in two dimensions the number of such neighbors is too small.

In Vicsek's model for boids, however, one gets order even in two dimensions. Why? Due to the persistent motion of the boids they interact over time with many more other boids, even with boids that at some point in time are quite far away. The interaction therefore attains effectively a much larger range, allowing $\eta_c > 0$.

In robotic applications it is also of interest to understand what happens when individual robots fail, i.e. persistently moves in a different direction than the swarm. Does the whole swarm get disturbed? Can it even break up with some other robots following the failing one?

# 4 Error Estimate and Time Step Control

Quality of code:

- validate the code
- assess error: does the code exhibit correct order of convergence?

No analytical solution: how do we estimate the accuracy of the numerical solution?

Make use of error assessment:

- for efficiency use estimate to adjust time step
- use estimate to improve solution

## 4.1 Validation of the Code

**No code should be used without validation!**

**No exact solution:**

For most problem that we address numerically we do not have an exact solution to compare to. How can we validate that we coded the problem correctly? In many problems one can consider special cases for which analytical solutions are available. Compare the code in those cases.

**Example:**

In the flocking homework problem one can test each of the three force terms individually

$$\mathbf{f_i} = \underbrace{\frac{\bar{\mathbf{v}}_i}{|\bar{\mathbf{v}}_i|}}_{\mathbf{F_1}} \underbrace{-\mathbf{v}_i}_{\mathbf{F_2}} \underbrace{- \sum_{j=1}^{N} (\mathbf{r}_i - \mathbf{r}_j) \left( \alpha e^{-\Delta_{ij}^2/\Delta_{att}^2} - \beta \frac{e^{-\Delta_{ij}^2/\Delta_{rep}^2}}{\Delta_{ij}^2 + \delta_{rep}^2} \right)}_{\mathbf{F_3}}.$$

Setting all terms except for one of them to 0 one should observe

- $\mathbf{F}_2$: all velocities should decay exponentially with a decay rate of 1
- $\mathbf{F}_1$& $\mathbf{F}_2$: the velocity of all particles should have magnitude 1
- $\mathbf{F}_3$: with only $\mathbf{F}_3$ present the equations become Newton's equation of motion with a central force between pairs of particles $i$ and $j$

  $$\mathbf{F}_3(\mathbf{r}_i, \mathbf{r}_j) = -(\mathbf{r}_i - \mathbf{r}_j) f(|\mathbf{r}_i - \mathbf{r}_j|)$$

  If, for simplicity, only two particles are considered and one particle is kept fixed by setting $\mathbf{f}_1 = 0$, then for suitable initial conditions the second particle will follow a circular trajectory around the first particle in which the centrifugal force is balanced by the attractive force.

**Notes:**

- If each term in the model has been tested successfully, one gains confidence that one has implemented the model correctly in the code.
- Of course, there could be situations in which the combination of terms can lead to new situations, which have not been tested. Therefore one always has to keep in mind that the code could still have bugs in it:
  - all results should be viewed with suspicion
  - one always needs to consider whether the results make sense.

**Convergence:**

Even if the code appears to generate a reasonable ('correct') solution it is important to check how accurate it actually is.

We have for the local truncation error

$$\tau_l = \mathcal{O}(\Delta t^p)$$

**Note:** remember the global error $\tau_g$ is expected to be $\mathcal{O}(\Delta t^{p-1})$.

1. Measure the error as a function of $\Delta t$ for a fixed time interal $T$

   $$\tau_g(\Delta t) = a\Delta t^{p-1}$$

   **Note:** decrease $\Delta t$ always by **at least a factor of 2!**

2. Extract $p$ using a log-log-plot

   $$\ln\, \tau_g = (p-1) \ln\, \Delta t + \ln a$$

   **Notes:**

   - in a log-log-plot $p-1$ is given by the slope of the curve
   - to measure slope check by how many decades $\tau_g$ goes down when $\Delta t$ is varied by a decade

For the code to be trustworthy:

1. The error $\tau_g$ has to go to 0 as $\Delta t$ goes to 0
2. The slope has to agree with the expected order of the method

**Demo Example:**

Consider differential equation

$$\frac{dy}{dt} = -\frac{1}{2y} \qquad 0 \le t \le T \qquad \text{with} \qquad y(0) = 1$$

It has the exact solution

$$y(t) = \sqrt{1-t}.$$

**Notes:**

- If the data points do not fall on a *convincingly straight* line it makes **no sense** to fit a straight line through them:
  rather, the conclusion is that for the parameters used the global error $\tau_g$ has not reached the asymptotic regime yet

  $$\tau_g = a\Delta t^{p-1} + b\Delta t^p + ...$$

  i.e. for the values of $\Delta t$ used the term $\mathcal{O}(\Delta t^p)$ is not negligible compared to $a\Delta t^{p-1}$.

- If the slope does not agree with the expected order then there is still an error in the code.
  Usually the slope is too small and as a result the coding effort and the additional computation time to obtain and run the refined method are wasted.

## 4.2 Error Estimate and Extrapolation

Typically, no analytical result is available for the solution we are interested in: how do we estimate its global error?

Let us compare approximations for two different values of the time step, $\Delta t$ and $\Delta t/2$ (at fixed final time $T$),

$$\begin{aligned} y_{\Delta t}(T) &= y_e(T) + a(T)\,\Delta t^{p-1} + ... \\ y_{\Delta t/2}(T) &= y_e(T) + a(T)\,(\frac{\Delta t}{2})^{p-1} + ... \\ \Rightarrow y_{\Delta t}(T) - y_{\Delta t/2}(T) &= a(T)\,\Delta t^{p-1}\left(1 - \frac{1}{2^{p-1}}\right) + \mathcal{O}(\Delta t^p) \end{aligned}$$

From this we can extract an estimate for the global error

$$y_{\Delta t}(T) - y_e(T) \sim a(T)\,\Delta t^{p-1} \sim \frac{1}{1-2^{1-p}}\,\left(y_{\Delta t}(T) - y_{\Delta t/2}(T)\right) \tag{9}$$

**Notes:**

- For validation plot $y_{\Delta t}(T) - y_{\Delta t/2}(T)$ double-logarithmically for at least 4 values of $\Delta t$ changing $\Delta t$ by factors of 2 and measure the slope
- If the change $y_{\Delta t}(T) - y_{\Delta t/2}$(T) shows the expected power law then the error is close to the change of the solution when $\Delta t$ is changed by a factor of 2: $\frac{1}{1-2^{1-p}} < 2$ since $p \geq 2$.
- This estimate neglects higher-order contributions to the error

**Richardson Extrapolation**

We can make use of our estimate (9) to obtain a better, higher-order approximation for $y_e(T)$

$$\begin{aligned} y_e(T) &= y_{\Delta t}(T) - a(T)\,\Delta t^{p-1} + \mathcal{O}(\Delta t^p) \\ &= y_{\Delta t}(T) - \frac{1}{1-2^{1-p}}\left(y_{\Delta t}(T) - y_{\Delta t/2}(T)\right) + \mathcal{O}(\Delta t^p) \\ &= \frac{1}{1-2^{1-p}}\left(y_{\Delta t/2}(T) - 2^{1-p} y_{\Delta t}(T)\right) + \mathcal{O}(\Delta t^p). \end{aligned}$$

**Notes:**

- For the extrapolated result one does not have an error estimate any more; the order of accuracy is known but we do not have an estimate for the prefactor (cf. eq.(9))
- The extrapolation may affect the stability of the algorithm. It can lead to difficulties in stiff problems.

## 4.3 Adaptive Time Step

Since we have estimates for the error we can use it to adapt the time step $\Delta t_n$ depending on the demands of the solution at a given time.

**Demo:**

$$\frac{dy}{dt} = -\frac{1}{4y^3} \qquad 0 \leq t \leq T \qquad \text{with} \qquad y(0) = 1 \tag{10}$$

has the exact solution

$$y = (1-t)^{\frac{1}{4}}$$

which becomes singular at $t = 1$, i.e. the derivative diverges. Near $t = 1$ very small time steps are required to resolve the steep gradients. During the initial phase large time steps would be sufficient and much more efficient. Problems like this are best solved using adaptive time steps.

To adapt each individual time step we need estimates for the local error $\tau_n$ incurred at time step $n$. Analogous to the estimate of the global error (9) we can estimate that error by comparing the errors when taking time steps of different size

$$\begin{aligned} y_{\Delta t} &= y_e + a\,\Delta t_n^p + \mathcal{O}(\Delta t_n^{p+1}) \\ y_{\Delta t/2} &= y_e + 2a\,(\frac{\Delta t_n}{2})^p + \mathcal{O}(\Delta t_n^{p+1}) \end{aligned}$$

where we assume that the errors incurred by taking two time steps of size $\frac{\Delta t}{2}$ each add up.

We would like to get the global error $\tau_g$ at the final time $T$ below the tolerance $\Delta_{tol}$,

$$\tau_g = \sum_{n=1}^{N} |\tau_n(\Delta t_n)| \leq \Delta_{tol}$$

A reasonable approach is then to require that the individual time steps are chosen such that the local errors $\tau_n$ satisfy

$$|\tau_n(\Delta t_n)| \leq \Delta_{tol} \frac{\Delta t_n}{T}.$$

If the local errors simply added up this would guarantee

$$\tau_g \leq \frac{\Delta_{tol}}{T} \sum_{n=1}^{N} \Delta t_n = \Delta_{tol}.$$

**Note:**

- Since we are looking at the local error the error goes like $\Delta t^p$ rather than $\Delta t^{p-1}$.

Solving for the local error we obtain

$$\tau_n(\Delta t_n) \equiv a\,\Delta t_n^p = \frac{1}{1-2^{1-p}}\,\left(y_{\Delta t} - y_{\Delta t/2}\right)$$

We can now compare the estimated local error $\tau_n(\Delta t_n)$ with the tolerance $\Delta_{tol} \cdot \Delta t_n/T$ and accept or reject this time step.

Optimally, the time step is chosen such that the error is equal to the allowed tolerance.

We pick the optimal time step $\Delta t^*$ that satisfies

$$|\tau_n(\Delta t^*)| = \frac{\Delta_{tol}}{T}\,\Delta t^*$$

To solve for $\Delta t^*$ we need to know the dependence of $\tau$ on $\Delta t$

$$\tau_n(\Delta t) \quad = \quad a\,\Delta t^p$$

This gives in general

$$\Delta t^* = \left(\frac{\Delta_{tol}}{|a|T}\right)^{\frac{1}{p-1}}$$

We get an approximation for $a$ from the estimated error $\tau_n(\Delta t_n)$

$$a = \frac{\tau_n(\Delta t_n)}{\Delta t_n^p}$$

Inserting this $a$ into the condition for $\Delta t^*$ we obtain

$$\Delta t^* = \left(\frac{\Delta}{T}\right)^{\frac{1}{p-1}} \left(\frac{\Delta t_n^p}{|\tau_n(\Delta t_n)|}\right)^{\frac{1}{p-1}} \tag{11}$$

**Algorithm:**

1. compute $y(t_n + \Delta t)$ using $\Delta t$ and $\Delta t/2$ as time steps $\Rightarrow y_{\Delta t}$ and $y_{\Delta t/2}$
2. calculate $\tau_n$ based on $y_{\Delta t}$ and $y_{\Delta t/2}$ and determine $\Delta t^*$
   - if $\Delta t^* > \Delta t$ accept $y_{\Delta t/2}$ as $y(t_n + \Delta t)$ and increment $t_n \Rightarrow t_n + \Delta t$
   - if $\Delta t^* < \Delta t$ **do not** accept $y_{\Delta t/2}$ as $y(t + \Delta t)$, **do not** increment $t$ instead keep $y(t)$ and continue from there
3. adjust timestep
   $$\Delta t \Rightarrow \min(0.9\Delta t^*, T - t)$$
4. go to 1.

**Notes:**

- Do not use the full optimal time step $\Delta t^*$. This reduces the risk that the next time step is rejected because the error estimate was slightly off. The factor 0.9 is an adjustable parameter.
- Sometimes it may be useful to not increase the time step too fast to avoid that the new time step turns out to be too large. You could limit the increase to a factor of 2, say.
- The local error is estimated to be below the tolerance when using the time step $\Delta t$; we have also the solution with $\Delta t/2$. Use that solution instead.
- We can use $y_{\Delta t}$ and $y_{\Delta t/2}$ to use Richardson extrapolation to get a higher-order approximation; we will loose a reliable error estimate.

- During phases when the solution changes more rapidly $|a|$ is larger and $\Delta t^*$ becomes smaller. This decrease is much more pronounced for small $p$ (forward Euler, say) than for larger $p$ (Runge-Kutta).

---

The estimate of the local error can be done more systematically without *assuming* that the errors add up:

Error taking a single time step of size $\Delta t$:

$$y_e(t_{j+1}) - y_{\Delta t}(t_{j+1}) = \psi(t_j, y(t_j))\Delta t^p + \mathcal{O}\left(\Delta t^{p+1}\right)$$

Here $\psi(t_j, y_j)$ is the *principal error function* and $y_{\Delta t}$ is the numerical solution starting at $t_j$ with $y_j$ and taking one regular time step $\Delta t$.

Taking two time steps of size $\Delta t/2$ we get

$$\begin{aligned} y_e(t_{j+1/2}) - y_{\Delta t/2}(t_{j+\frac{1}{2}}) &= \psi\left(t_j, y(t_j)\right)\left(\frac{\Delta t}{2}\right)^p + \mathcal{O}\left(\Delta t^{p+1}\right) \\ \tilde{y}_e(t_j) - y_{\Delta t/2}(t_j) &= \underbrace{\psi\left(t_{j+\frac{1}{2}}, y_{\Delta t/2}(t_{j+\frac{1}{2}})\right)}_{\psi(t_j, y(t_j)) + \mathcal{O}(\Delta t)}\left(\frac{\Delta t}{2}\right)^p + \mathcal{O}\left(\Delta t^{p+1}\right) \end{aligned}$$

Here $\tilde{y}_e(t_j)$ is the exact solution at $t_j$ that started at $t_{j+\frac{1}{2}}$ with the value $y_{\Delta t/2}(t_{j+\frac{1}{2}})$ obtained by the algorithm in the previous step . Thus, in general $\tilde{y}_e(t) \neq y_e(t)$

We would like to obtain an estimate for the error by comparing $y_{\Delta t}(t_{j+1})$ and $y_{\Delta t/2}(t_{j+1})$. Consider therefore

$$\left(y_e(t_{j+1}) - y_{\Delta t}(t_{j+1})\right) - \left(\tilde{y}_e(t_{j+1}) - y_{\Delta t/2}(t_{j+1})\right) = \left(1 - 2^{-p}\right)\psi(t_j, y(t_j))\Delta t^p + \mathcal{O}(\Delta t^{p+1}).$$

We need to compare $y_e$ and $\tilde{y}_e$,

$$y_e(t_{j+1}) - \tilde{y}_e(t_{j+1}) = y_e(t_{j+\frac{1}{2}}) - \tilde{y}_e(t_{j+\frac{1}{2}}) + \mathcal{O}\left(\frac{\Delta t}{2}\frac{d}{dt}\left(y_e(t_{j+\frac{1}{2}}) - \tilde{y}_e(t_{j+\frac{1}{2}})\right)\right)$$

The exact solutions $\tilde{y}_e$ and $y_e$ both satisfy the differential equation

$$\frac{dy}{dt} = F(t, y)$$

where we assume $F$ is sufficiently smooth. More precisely, we require that $F$ is Lipschitz continuous, i.e.

$$\|F(t, y_1) - F(t, y_2)\| \leq L\|y_1 - y_2\| \qquad \text{for any } y_1 \text{and } y_2$$

with the Lipschitz constant $L$.

Therefore

$$\mathcal{O}\left(\frac{\Delta t}{2}\frac{d}{dt}(\tilde{y}_e(t) - y_e(t))\right) \leq \mathcal{O}\left(\frac{\Delta t}{2}L\left\|\tilde{y}_e(t_{j+\frac{1}{2}}) - y_e(t_{j+\frac{1}{2}})\right\|\right)$$

and

$$\left\|y_e(t_{j+\frac{1}{2}}) - \tilde{y}_e(t_{j+\frac{1}{2}})\right\| = \left\|y_e(t_{j+\frac{1}{2}}) - y_{\Delta t/2}(t_{j+\frac{1}{2}})\right\| = \left|\psi\left(t_j, y(t_j)\right)\right|\left(\frac{\Delta t}{2}\right)^p + \mathcal{O}\left(\Delta t^{p+1}\right)$$

We therefore have

$$y_e(t_{j+1}) - \tilde{y}_e(t_{j+1}) = y_e(t_{j+\frac{1}{2}}) - \underbrace{\tilde{y}_e(t_{j+\frac{1}{2}})}_{y_{\Delta t/2}(t_{j+\frac{1}{2}})} + \mathcal{O}\left(\Delta t^{p+1}\right) = \psi\left(t_j, y(t_j)\right)\left(\frac{\Delta t}{2}\right)^p + \mathcal{O}\left(\Delta t^{p+1}\right).$$

Thus

$$y_{\Delta t/2}(t_{j+1}) - y_{\Delta t}(t_{j+1}) = \left(1 - 2^{-p} - 2^{-p}\right) \underbrace{\psi(t_j, y(t_j))\, \Delta t^p}_{\tau(\Delta t)} + \mathcal{O}\left(\Delta t^{p+1}\right)$$

Solving for the local error we get as before

$$\tau(\Delta t) = \frac{1}{1 - 2^{1-p}} \left(y_{\Delta t/2}(t_{j+1}) - y_{\Delta t}(t_{j+1})\right).$$

---

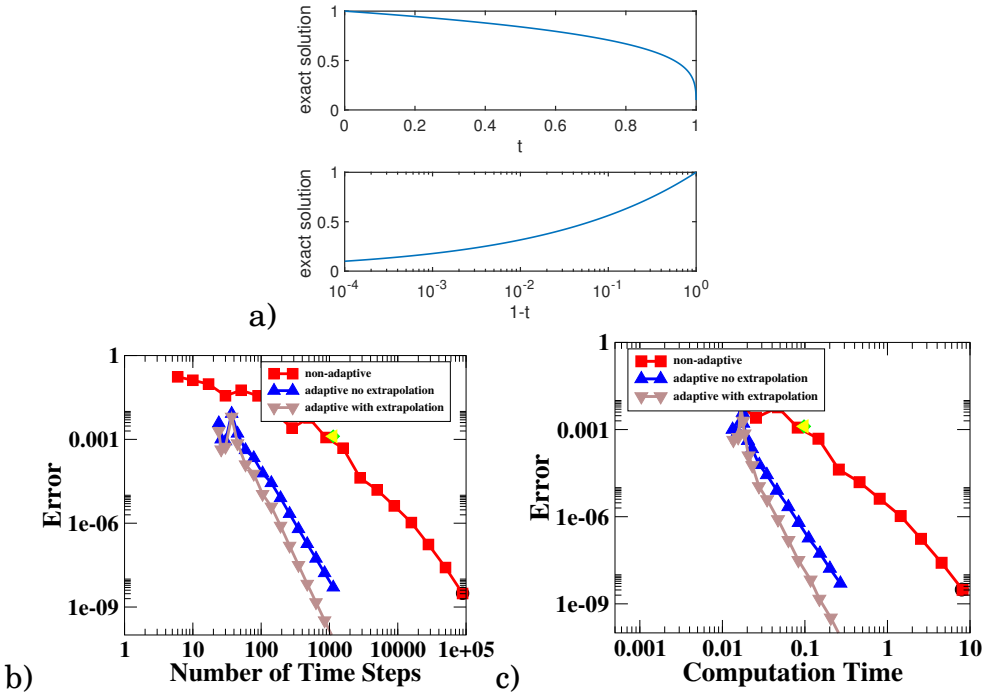



Figure 3: a) Exact solution in two representations. Error as a function of number of time steps (b) and as a function of computation time (c) for RK4 (non-adaptive, adaptive without Richardson, and with Richardson extrapolation.) Note that the computation time per time step is smaller for the non-adaptive scheme than for the adaptive one (3 evaluations per time step). $dy/dt = -1/4y^3$, error at $t = 0.9999$ with initial condition $y(0) = 1$.

**Note:**

- It turns out that for (10) forward Euler and backward Euler do not gain from using an adaptive time step: the computation time with adaptive time step is in fact larger than without.

Even though the local error estimates are quite good, their sum does not match the true error arising in the simulation (Fig.4).

We assumed that the global error is simply the sum of the local errors. However, this is in general not the case. An error made at time $t_n$ can be amplified (or reduced) by the subsequent evolution of the system (Fig.5).

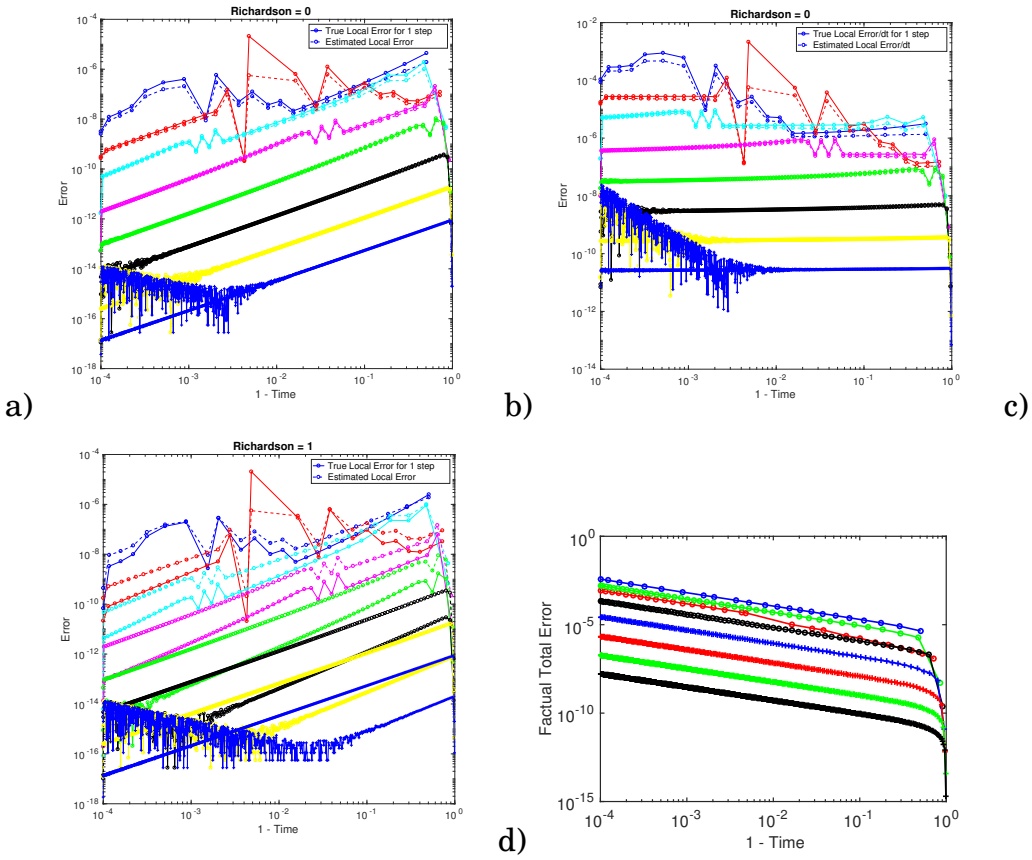



Figure 4: a) The estimate of the local error agrees well with the true local error (without Richardson extrapolation). b) THe adaptive time step keeps the estimated local error divided by $\Delta t$ quite constant. c) with Richardson extrapolation the true local error is much smaller than the estimated local error. d) the factual error increases steadily. .

**Note:**

- Instead of specifying a tolerance $\Delta$ for the absolute local error $\tau_n$ one can also specify a tolerance $\Delta_{rel}$ for the relative local error

$$\tau_{n,rel} = \frac{\tau_n}{|y_n|}.$$

The optimal time step is then given by an analogous expression

$$\Delta t^* = \left(\frac{\Delta_{rel}}{T}\right)^{\frac{1}{p-1}} \left(\frac{\Delta t_n^p}{|\tau_{n,rel}(\Delta t_n)|}\right)^{\frac{1}{p-1}}.$$

- If the solution changes sign a condition on the relative error may be too restrictive. One could replace the condition on the relative error by one on the absolute error when the magnitude of the solution becomes too small.

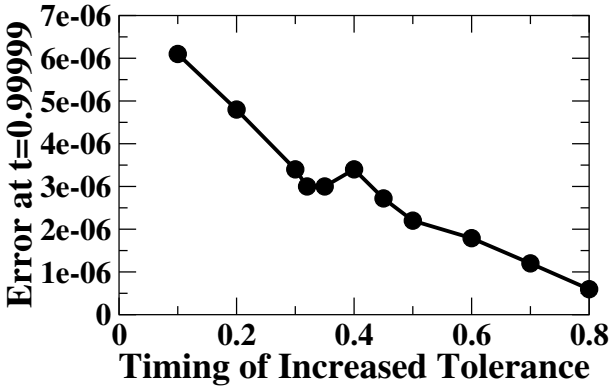



Figure 5: The global error depends on when the local error is incurred. In (10) the global error is larger when the tolerance is increased early in the simulation. In the plot the tolerance is increased by a factor of 50 for a duration of $\Delta t = 0.2$ and the error is measured at $T = 0.99999$ for $dy/dt = -1/4y^3$.

- For systems the errors in each of the components can be monitored and controled. One could require the maximal absolute error or the maximal relative error to be below some tolerance.

**Pair Method for Error Estimate**

We can also estimating the error by computing $y_{n+1}$ with two methods with different order of accuracy

$$\begin{aligned} y_{n+1} &= y_n + a\Delta t^p + b\Delta t^{p+1} \\ y^*_{n+1} &= y_n + b'\Delta t^{p+1}. \end{aligned}$$

Then the leading error of $y_{n+1}$ is $a\Delta t^p$

$$y_{n+1} - y^*_{n+1} = a\Delta t^p + (b - b')\Delta t^{p+1}$$

and the estimate for the local error is given by

$$\tau_n = a\Delta t^p = y_{n+1} - y^*_{n+1} + \mathcal{O}(\Delta t^{p+1})$$

**Note:**

- Matlab *ode45* uses a pair RK4+RK5 for its error estimation (Dormand-Prince). The method involves 7 stages and both methods (RK4 and RK5) use the same intermediate time steps and auxiliary variable, i.e. they have the same coefficients $\alpha_k$ and $\beta_{mk}$ in the Butcher array.

**Backward-Forward Error Estimate**

Another possibility to obtain an error estimate is to use the same method with the same time step, but integrating once forward and once backward in time.

Introduce the exact solution $\tilde{y}(t)$ starting from $y_n$ at $t_n$, $\tilde{y}(t_n) = y_n$, and the exact solution $\tilde{y}^*(t)$, which starts at time $t_{n+1}$ at $y_{n+1}$ and goes backward in time $\tilde{y}^*(t_{n+1}) = y_{n+1}$.

$$\begin{aligned} y_{n+1} - \tilde{y}(t_{n+1}) &= a(t_n, y_n)\Delta t^p + \mathcal{O}\left(\Delta t^{p+1}\right) \\ y_n^* - \tilde{y}^*(t_n) &= a(t_{n+1}, y_{n+1})\left(-\Delta t\right)^p + \mathcal{O}\left(\Delta t^{p+1}\right) \\ &= \left(a(t_n, y_n) + \mathcal{O}\left(\Delta t\right)\right)\left(-\Delta t\right)^p + \mathcal{O}\left(\Delta t^{p+1}\right). \end{aligned}$$

$y_n^*$ denotes the numerical value obtained by the backward step. For $p$ even we can solve for the truncation error by adding the two equations

$$\begin{aligned} 2a(t_n, y_n)\Delta t^p &= \underbrace{y_{n+1}}_{=\tilde{y}^*(t_{n+1})} - \tilde{y}(t_{n+1}) + y_n^* - \tilde{y}^*(t_n) + \mathcal{O}(\Delta t^{p+1}) \\ &= \tilde{y}^*(t_{n+1}) - \tilde{y}(t_{n+1}) + y_n^* - \tilde{y}^*(t_n) + \mathcal{O}(\Delta t^{p+1}). \end{aligned}$$

We would like to have an expression involving $y_n^* - y_n$, which is the same as $y_n^* - \tilde{y}(t_n)$. We therefore want to replace the terms involving the exact solutions at $t_{n+1}$ by terms evaluated at $t_n$. The exact solutions satisfy

$$\frac{d\tilde{y}}{dt} = F(t, \tilde{y}) \qquad \frac{d\tilde{y}^*}{dt} = F(t, \tilde{y}^*)$$

where we assume $F(t, y)$ is nice and smooth[11]. Therefore we can use a Taylor expansion around $t_{n+1}$

$$\tilde{y}^*(t_n) - \tilde{y}(t_n) = \tilde{y}^*(t_{n+1}) - \Delta t F(t_{n+1}, \tilde{y}^*(t_{n+1})) - \{\tilde{y}(t_{n+1}) - \Delta t F(t_{n+1}, \tilde{y}(t_{n+1})\} + \mathcal{O}(\Delta t^2).$$

The Taylor expansion is performed for the same function, but at slightly different expansion points: $\tilde{y}^*(t_{n+1})$ vs $\tilde{y}(t_{n+1})$. However,

$$\tilde{y}^*(t_{n+1}) - \tilde{y}(t_{n+1}) = \mathcal{O}\left(\Delta t^p\right).$$

Therefore we can estimate

$$\begin{aligned} \tilde{y}^*(t_n) - \tilde{y}(t_n) &= \tilde{y}^*(t_{n+1}) - \tilde{y}(t_{n+1}) - \Delta t \underbrace{\left[F(t_{n+1}, \tilde{y}^*(t_{n+1})) - F(t_{n+1}, \tilde{y}(t_{n+1}))\right]}_{\mathcal{O}(L \cdot |\tilde{y}^*(t_{n+1}) - \tilde{y}(t_{n+1})|)} \\ &= \tilde{y}^*(t_{n+1}) - \tilde{y}(t_{n+1}) - \mathcal{O}\left(\Delta t L \underbrace{|y_{n+1} - \tilde{y}(t_{n+1})|}_{\mathcal{O}(\Delta t^p)}\right) = \tilde{y}^*(t_{n+1}) - \tilde{y}(t_{n+1}) + \mathcal{O}(\Delta t^{p+1}) \end{aligned}$$

For the higher-order terms of the Taylor expansion, $\mathcal{O}\left(\Delta t^2\right)\ldots$, a similar argument holds.

Inserting this into the expression for the error we get

$$\tau_n = a(t_n, y_n)\Delta t^p = \frac{1}{2}\left(y_n^* - y_n\right) + \mathcal{O}\left(\Delta t^{p+1}\right).$$

**Note:**

[11] Actually, only Lipschitz continuity is needed.

- Intuitively, the result makes sense: we obtained $y_n^*$ from $y_n$ by first taking a step forward, which incurred an error of $\Delta t^p$, and then we took a step backward, which incurred almost the same error again. The two errors accumulated because $p$ is even.

# 5 Stability and Convergence

At each time step errors arise:

- truncation error
- computer makes round-off errors

Question: do the errors grow **catastrophically** or do they lead only to a finite modification of the solution? Catastrophic growth occurs if the scheme is **unstable.**

**Demo:**

$$\frac{dy}{dt} = \lambda y + A\sin(\omega t), \qquad \text{with } \lambda < 0, \qquad y(0) = y_0 = -A\frac{\omega}{\lambda^2 + \omega^2}$$

Analytical particular solution:

$$y(t) = \frac{A}{\lambda^2 + \omega^2}\left(-\lambda \sin \omega t - \omega \cos \omega t\right)$$

General solution

$$y(t) = ce^{\lambda t} + \frac{A}{\lambda^2 + \omega^2}\left(-\lambda \sin \omega t - \omega \cos \omega t\right)$$

Plug in:

$$\frac{dy}{dt} - \lambda y = \frac{A}{\lambda^2 + \omega^2}\left(-\lambda\omega \cos \omega t + \omega^2 \sin \omega t + \lambda^2 \sin \omega t + \lambda\omega \cos \omega t\right) = A \sin \omega t$$

**Note:**

- Homogeneous solution
$$y(t) = ae^{\lambda t}$$
for large $-\lambda > 0$ there is a very rapid decay towards the particular solution, which would need to be resolved with sufficiently small $\Delta t$.
- Even if $a = 0$ the decay rate $\lambda$ limits the maximal time step for explicit schemes

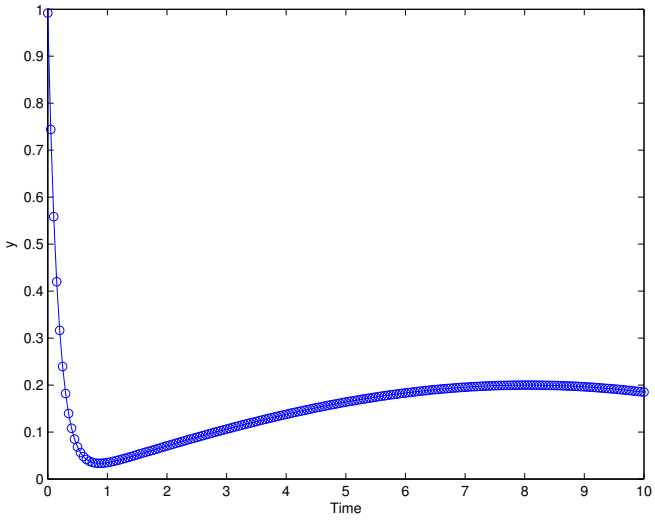



Figure 6: Example of the evolution in a stiff problem: after a fast initial transient the system evolves slowly, but the stability of the numerical solution is still governed by the fast time scale.

**Explore simple case:**
Forward Euler for

$$\frac{dy}{dt} = \lambda y \qquad \text{with} \qquad \lambda \leq 0$$

The exact solution $\tilde{y}(t) = y_0 e^{\lambda t}$ implies

$$\tilde{y}(t_n) = y_0 \left(e^{\lambda \Delta t}\right)^n$$

Since $\lambda \leq 0$ the exact solution does not grow and does not blow up.

The numerical solution

$$y_{n+1} = y_n + \Delta t \, (\lambda y_n) = (1 + \Delta t \, \lambda) \, y_n$$

implies

$$y_n = (1 + \Delta t \, \lambda)^n \, y_0$$

In order for the numerical solution not to diverge we need

$$|1 + \Delta t \, \lambda| \leq 1 \qquad \Leftrightarrow \qquad -1 \leq 1 + \Delta t \, \lambda \leq +1$$

Thus, the numerical scheme leads to spurious growth if $\Delta t$ is too large!
For stability we need

$$\Delta t \leq \frac{2}{|\lambda|}.$$

Most problems that have more than one time scale, e.g.,

- Time-varying coefficients that vary on a different time scale than the other characteristic times of the differential equation (e.g. $\lambda$)

$$\frac{du}{dt} = -\lambda u + A\cos\omega t$$

- Coupled ODEs

$$\begin{aligned} \frac{du}{dt} &= \lambda_1 u + uv - u^3 \\ \frac{dv}{dt} &= \lambda_2 v + uv - v^3 \end{aligned}$$

  have two different time scales (here $\lambda_{1,2}$)

- Partial differential equations can be thought of as many coupled ODEs.

In general, the time step has to be chosen to resolve all time scales involved, e.g. in the examples above

$$\Delta t \ll \lambda_i \qquad \Delta t \ll \frac{2\pi}{\omega}.$$

However, often the actual solution does not exhibit temporal evolution at all time scales at all times, e.g. the evolution could become slow after an initial rapid decay (cf. Fig.6):

- Initial phase: the accuracy requires resolving the 'boundary layer': small $\Delta t$ is unavoidable

- Later phase: the accuracy would not require small $\Delta t$
  **But:** stability may require small time steps$\Delta t$, the size of which is determined by the short time scale of the boundary layer.

**Definition:**

If a system has disparate time scales and the desired solution exhibits long periods during which its variation does not reflect the fast time scales the system is called *stiff.*

**Note:**

- To treat stiff problems efficiently, the method must have a large stability region or be *unconditionally stable* $\Rightarrow$ use implicit methods

Consider now more generally a linear system with $N$ components

$$\frac{d}{dt}\mathbf{y} = \mathbf{L}\,\mathbf{y}.$$

In many cases $\mathbf{L}$ can be diagonalized

$$\mathbf{SLS}^{-1} = \Lambda \qquad \frac{d}{dt}\mathbf{Sy} = \mathbf{SLS}^{-1}\mathbf{Sy} = \Lambda\mathbf{Sy}$$

with

$$\mathbf{\Lambda} = \begin{pmatrix} \lambda_1 & & & \\ & \lambda_2 & & \\ & & \ldots & \\ & & & \lambda_N \end{pmatrix}$$

and $\lambda_i \in \mathbb{C}$ being the eigenvalues of $L$.

Then it is sufficient to consider the *scalar* equation

$$\frac{d}{dt}y = \lambda y \qquad \lambda \in \mathbb{C}.$$

Meaning of complex $\lambda$: oscillations

Consider

$$\begin{pmatrix} \frac{du}{dt} \\ \frac{dv}{dt} \end{pmatrix} = L \begin{pmatrix} u \\ v \end{pmatrix} = \begin{pmatrix} 0 & 1 \\ -1 & 0 \end{pmatrix} \begin{pmatrix} u \\ v \end{pmatrix}$$

The eigenvalues of $L$ are given by

$$\lambda^2 + 1 = 0 \qquad \Rightarrow \lambda = \pm i$$

with the associated eigenvectors

$$\mathbf{V}^{(1,2)} = \begin{pmatrix} 1 \\ \pm i \end{pmatrix}$$

The general solution is then given by

$$\begin{pmatrix} u \\ v \end{pmatrix} = A \begin{pmatrix} 1 \\ i \end{pmatrix} e^{it} + A^* \begin{pmatrix} 1 \\ -i \end{pmatrix} e^{-it} = \begin{pmatrix} 2R\cos(t+\phi) \\ -2R\sin(t+\phi) \end{pmatrix} \qquad \text{with} \qquad A = Re^{i\phi}$$

**Definition:**

A numerical scheme for the differential equation

$$\frac{dy}{dt} = \lambda y \qquad \lambda \in \mathbb{C} \quad \text{with} \quad \lambda_r \leq 0$$

is called *absolutely stable* if it corresponds to an iteration

$$y_n = y_0\, z^n \qquad \text{with} \qquad z \in \mathbb{C} \qquad \text{and} \qquad |z| \leq 1.$$

**Definition:**

The *region of absolute stability* of a numerical scheme is given by the region $A$ in the complex plane defined by

$$A = \{\lambda\Delta t \in \mathbb{C} \,|\, \|y\| \text{ bounded for all } t\}\,.$$

**Notes:**

- If L has eigenvalues $\lambda$ with positive real part, $\lambda_r > 0$, even the analytical solutions will diverge: definition of absolute stability is only meaningful if $\lambda_r \leq 0$ for *all* eigenvalues of L.
- The stability condition can be expressed as:
  *the method does not amplify small differences* (e.g. when starting from slightly different initial values $y_0$).
  Thus, the truncation errors made at an earlier time do not get amplified exponentially with time to destroy the solution over time.

**Stability Analysis:**

Use the Ansatz

$$y_n = y_0 z^n \qquad \text{with} \qquad z \in \mathbb{C}.$$

Insert it in the numerical scheme and determine $z$:

$$|z| \leq 1 \quad \Rightarrow \qquad y \text{ bounded for all times} \qquad \text{scheme stable}$$

$$|z| > 1 \quad \Rightarrow y \to \infty \qquad \text{for} \qquad n \to \infty \qquad \text{scheme unstable}$$

**Examples:**

1. Adams-Bashforth Methods

   (a) Forward Euler:

   $$y_{n+1} = y_n + \lambda\Delta t \qquad z = 1 + \lambda\Delta t = 1 + \lambda_r\Delta t + i\lambda_i\Delta t$$

   thus

   $$|z|^2 = (1 + \lambda_r\Delta t)^2 + \lambda_i^2\Delta t^2$$

   The stability limit is a circle in the complex $\lambda\Delta t-$plane.

   For $\lambda \in \mathbb{R}$ (non-oscillatory systems) the stability limit is given by $\Delta t = 2/|\lambda_r|$.

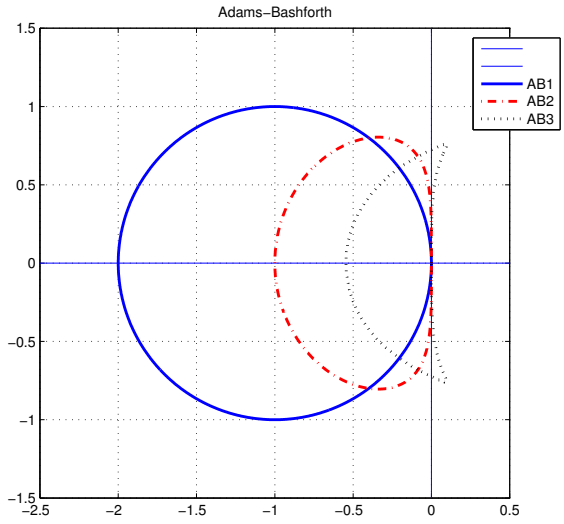



Figure 7: The regions of absolute stability for Adams-Bashforth methods.

- The instability is characterized by $z \leq -1 < 0$:
  The instability induces oscillations in which the solution changes sign in *each* time step: this oscillation has nothing to do with the dynamics of the differential equation, it is completely due to the numerical scheme (cf. demo simulations). This is a typical signature of a catastrophic numerical instability.

For $\lambda \in i\mathbb{R}$ (oscillatory systems) forward Euler is unstable for *all* $\Delta t$.

- The instability is mild: the growth rate goes to 0 with $\Delta t \to 0$ (the imaginary axis is tangential to the stability limit at $\lambda \Delta t = 0$).

(b) AB2:

$$y_{n+1} = y_n + \Delta t \lambda \left( \frac{3}{2} y_n - \frac{1}{2} y_{n-1} \right) \qquad z = 1 + \Delta t \lambda \left( \frac{3}{2} - \frac{1}{2} \frac{1}{z} \right)$$

There is a simple method to plot the stability limits:
Insert $z = e^{i\theta}$ into the equation for $z$:

$$\cos\theta + i \sin\theta = 1 + \Delta t(\lambda_r + i\lambda_i) \left( \frac{3}{2} - \frac{1}{2} (\cos\theta - i \sin\theta) \right)$$

and solve for $\lambda_r \Delta t$ and $\lambda_i \Delta t$ as a function of $\theta$ to get the curve for the stability limit parametrized by $\theta$.
The stability region for AB2 is smaller than for FE. AB2 also unstable for $\lambda \in i\mathbb{R}$, but again the growth rate goes to 0 for $\Delta t \to 0$. In fact, it goes to 0 faster than for FE; instability more mild than for FE.
Third order: also stable for oscillatory systems, stability region includes

a range with $\lambda \in i\mathbb{R}$.
Stability region shrinks with increasing order of the scheme.

2. Adams-Moulton

   (a) AM1=BE

   $$y_{n+1} = y_n + \Delta t\lambda\, y_{n+1}$$
   $$y_{n+1} = \frac{1}{1-\Delta t\lambda}\, y_n$$

   With $\Delta t\lambda = \Delta t(\lambda_r + i\lambda_i)$ we get

   $$|z|^2 = \frac{1}{(1-\Delta t\lambda_r)^2 + \Delta t^2\lambda_i^2}$$

   For $\lambda_r \leq 0$ we have $|z| < 1$ for *any* $\Delta t$. Thus, backward Euler is *unconditionally* stable for time steps of arbitrary size

   (b) AM2=Crank-Nicholson:

   $$\begin{aligned} y_{n+1} &= y_n + \frac{1}{2}\Delta t\lambda\,(y_{n+1} + y_n) \\ (1 - \frac{1}{2}\Delta t\lambda)\, y_{n+1} &= (1 + \frac{1}{2}\Delta t\lambda)\, y_n \end{aligned}$$

   thus

   $$\begin{aligned} z &= \frac{2+\Delta t\lambda}{2-\Delta t\lambda} = \frac{2+\Delta t\lambda_r + i\Delta t\lambda_i}{2-\Delta t\lambda_r - i\Delta t\lambda_i} \\ |z|^2 &= \frac{(2-\Delta t|\lambda_r|)^2 + \Delta t^2\lambda_i^2}{(2+\Delta t|\lambda_r|)^2 + \Delta t^2\lambda_i^2} \leq 1 \qquad \text{for } \lambda_r < 0 \end{aligned}$$

   Thus:
   Crank-Nicholson is also *unconditionally stable*, i.e. stable for all $\Delta t$.
   Of course, the accuracy will set a limit for the time step.
   **Note:**

   - AM3 and higher-order AM schemes are *not* unconditionally stable. Their stability regions shrink with increasing order.

3. Runge-Kutta

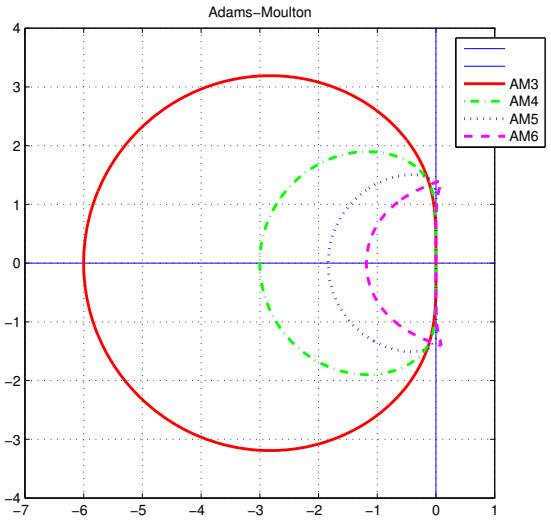



Figure 8: Regions of absolute stability for Adams-Moulton methods. BE and AM2=CN are unconditionally stable and no stability limits are appear in the $\lambda\Delta t$-plane.

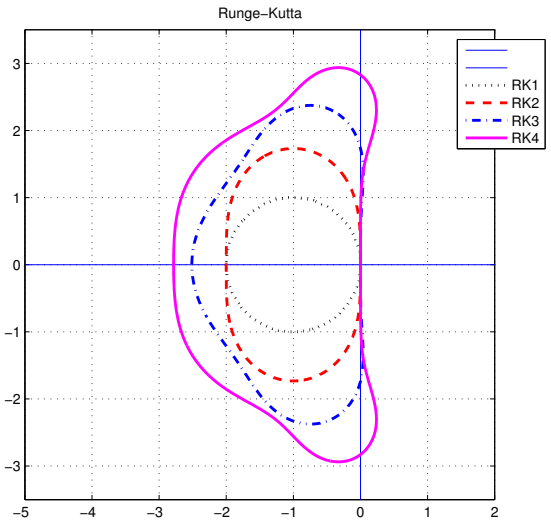



Figure 9: Region of absolute stability for Runge-Kutta methods.

**Notes:**

- For Runge-Kutta schemes the region of stability grows with increasing order.
- fourth-order Runge-Kutta stable for $\lambda \in \mathbb{R}$ and for $\lambda \in i\mathbb{R}$
  - RK4 is a very good all-purpose scheme

**General Connection between Stability and Convergence**

**Definition:**

A one-step scheme

$$y_{n+1} = y_n + \Delta t \mathcal{F}(t_n, y_n, \Delta t)$$

is called *consistent* for the differential equation $dy/dt = F(t, y)$ if

$$\lim_{\Delta t \to 0} \mathcal{F}(t_n, y_n, \Delta t) = F(t_n, y_n)$$

**Theorem:**

A scheme that is *consistent* and *stable* converges to the exact solution in the limit $\Delta t \to 0$.

**Note:**

- in each time step truncation and round-off errors arise. We need that they do not accumulate catastrophically, but remain bounded.

# 6 Implementation of Implicit Methods: Newton's Method

For nonlinear problems implicit methods require, in principle, to solve a nonlinear equation at each time step .

E.g. Crank-Nicholson for

$$\frac{dy}{dt} = F(t, y)$$

reads

$$y_{n+1} \quad = \quad y_n + \frac{1}{2}\Delta t \left(F(t_{n+1}, y_{n+1}) + F(t_n, y_n)\right)$$

We need to solve a nonlinear equation for $u \equiv y_{n+1}$

$$u = y_n + \frac{1}{2}\Delta t \left(F(t_{n+1}, u) + F(t_n, y_n)\right)$$

i.e., we need to solve

$$G(u) = 0 \qquad \text{with} \qquad G(u) = y_n + \frac{1}{2}\Delta t \left(F(t_{n+1}, u) + F(t_n, y_n)\right) - u. \tag{12}$$

In general, the most efficient approach is using *Newton's method*:

Extrapolate iteratively to $G(u) = 0$ by expanding around the current iterate (Fig.10):

$$\text{We want} \qquad G(u^{(l+1)}) = 0 \qquad \Rightarrow \qquad G(u^{(l)}) + (u^{(l+1)} - u^{(l)}) \left.\frac{dG}{du}\right|_{u^{(l))}} = 0.$$

Thus, the iteration is given by

$$u^{(l+1)} = u^{(l)} - \left( \left.\frac{dG(u)}{du}\right|_{u^{(l))}} \right)^{-1} G(u^{(l)}) \tag{13}$$

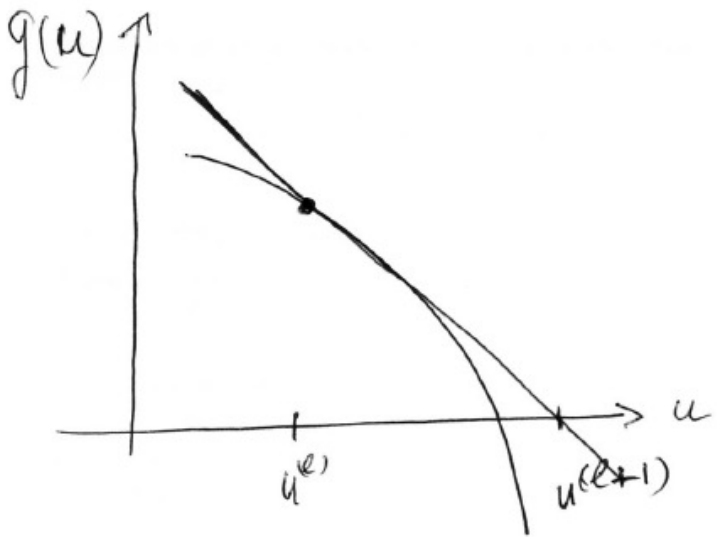



Figure 10: Sketch of Newton's method: extrapolate to the 0 of the function $G(u)$.

The Newton iteration converges very fast if the initial guess is sufficiently close to the solution $u^\infty$, i.e. if the initial guess is in the basin of attraction of the fixed point.

To see this, expand

$$u^{(l)} = u^\infty + \epsilon^{(l)}$$

and insert this into the Newton iteration

$$
\begin{aligned}
u^\infty + \epsilon^{(l+1)} &= u^\infty + \epsilon^{(l)} - \frac{G(u^\infty + \epsilon^{(l)})}{G'(u^\infty + \epsilon^{(l)})} \\
\epsilon^{(l+1)} &= \epsilon^{(l)} - \frac{\overbrace{G(u^\infty)}^{=0} + G'(u^\infty)\epsilon^{(l)} + \frac{1}{2}G''(u^\infty)\left(\epsilon^{(l)}\right)^2}{G'(u^\infty) + \epsilon^{(l)}G''(u^\infty)} \\
&= \epsilon^{(l)} - \epsilon^{(l)}\frac{1}{1 + \epsilon^{(l)}\frac{G''(u^\infty)}{G'(u^\infty)}} - \left(\epsilon^{(l)}\right)^2 \frac{1}{2}\frac{G''(u^\infty)}{G'(u^\infty)}\frac{1}{1 + \epsilon^{(l)}\frac{G''(u^\infty)}{G'(u^\infty)}} \\
\epsilon^{(l+1)} &= \frac{1}{2}\frac{G''(u^\infty)}{G'(u^\infty)}\left(\epsilon^{(l)}\right)^2 + \mathcal{O}\left(\left(\epsilon^{(l)}\right)^3\right)
\end{aligned}
$$

**Definition**

The order of the convergence of an iteration method is $p$ if the limit

$$\lim_{l\to\infty} \frac{\left|\epsilon^{(l+1)}\right|}{\left|(\epsilon^{(l)})^p\right|} = r$$

exists and is non-zero.

**Notes:**

- For linear convergence, $p = 1$, the approach to the fixed point is exponential

$$\epsilon^{(n)} \propto \epsilon^{(0)}\, r^n$$

- For quadratic convergence, $p = 2$, the approach to the fixed point is much faster than exponential

$$\epsilon^{(l+1)} = r\left(\epsilon^{(l)}\right)^2 = r\left(r\left(\epsilon^{(l-1)}\right)^2 r\left(\epsilon^{(l-1)}\right)^2\right)$$

$$
\begin{aligned}
\epsilon^{(1)} &= r\left(\epsilon^{(0)}\right)^2 \\
\epsilon^{(2)} &= r\left(\epsilon^{(1)}\right)^2 = r\left(r\left(\epsilon^{(0)}\right)^2 r\left(\epsilon^{(0)}\right)^2\right) \\
\epsilon^{(3)} &= r\left(r\left(r\left(\epsilon^{(0)}\right)^2 r\left(\epsilon^{(0)}\right)^2\right) r\left(r\left(\epsilon^{(0)}\right)^2 r\left(\epsilon^{(0)}\right)^2\right)\right)
\end{aligned}
$$

Thus, in each iteration the number of *factors* $\epsilon^{(0)}$ is doubled and the number $n^{(l)}$ of factors $r$ satisfies

$$n^{(l+1)} = 2n^{(l)} + 1$$

i.e.

$$
\begin{aligned}
n^{(1)} &= 1 \\
n^{(2)} &= 2n^{(1)} + 1 = 3 \\
n^{(3)} &= 2\left(2 \cdot 1 + 1\right) + 1 = 7 \\
n^{(4)} &= 2\left(2\left(2 \cdot +1\right) + 1\right) + 1 = 15
\end{aligned}
$$

$$n^{(n)} = \sum_{l=0}^{n-1} 2^l = 2^n - 1$$

Combined we have then

$$\epsilon^{(n)} = r^{-1} \left(r\epsilon^{(0)}\right)^{2^n}.$$

- For example for $r\epsilon^{(0)} = 0.1$:
  while for linear convergence the number of correct digits increases by one in each step, it *doubles* in each step for quadratic convergence

| | | | | | | |
|---|---|---|---|---|---|---|
| exponential | $10^{-1}$ | $10^{-2}$ | $10^{-3}$ | $10^{-4}$ | $10^{-5}$ | $10^{-6}$ |
| super-exponential | $10^{-1}$ | $10^{-2}$ | $10^{-4}$ | $10^{-8}$ | $10^{-16}$ | $10^{-32}$ |

**Notes:**

- In principle, in each time step (13) has to be iterated until $u^{(l)}$ converges to $u^* \equiv \lim_{l\to\infty} u_l$ which then gives $y_{n+1}$
- for $N$ coupled equations

$$\frac{dG}{du} \qquad \Rightarrow \qquad \text{Jacobian} \quad J_{ij}(u^{(l)}) = \frac{\partial G_i(u^{(l)})}{\partial u_j}$$

  the Jacobian depends on $u^{(l)} \Rightarrow$ it needs to be evaluated in each iteration:
  this can be *expensive* since the Jacobian matrix is $N \times N$ and can be large.

- Instead of inverting the Jacobian $J_{ij}$ it is better to multiply (13) by J and solve it in the form

$$\sum_j J_{ij}(u_1^{(l)}, ..., u_N^{(l)}) \left(u_j^{(l+1)} - u_j^{(l)}\right) = -G_i(u_1^{(l)}, ..., u_N^{(l)}).$$

  Solving $N$ linear equations takes $\mathcal{O}(N^2)$ operations, while inverting the matrix requires $\mathcal{O}(N^3)$ operations.
  In matlab this takes the form

$$\mathbf{u}^{(l+1)} - \mathbf{u}^{(l)} = -\mathbf{J} \backslash \mathbf{G}(\mathbf{u}^{(l)})$$

- The Jacobian may not be available analytically. In that case we need to determine it numerically

$$\frac{\partial G_j}{\partial u_i} \approx \frac{1}{\delta} \left(G_j(u_1, u_2, ...u_i + \delta, ...u_N) - G_j(u_1, ..., u_N)\right)$$

  where $\delta \neq 0$ is a small scalar number. In general, this needs to be done for each pair $(i, j)$.

- For time-dependent functions $F$(t,u) the Jacobian has to be determined at $t = t_{n+1}$ since $F(t, u)$ is evaluated at that time (cf. (12) ).

- When $\Delta t$ is not too large, $y_n$ is usually close to $y_{n+1}$ and $y_n$ is already a good initial guess.
  Often it is then sufficient to iterate only once to get close enough to $u^*$:

$$y_{n+1} = y_n - \left( \left. \frac{dG(u)}{du} \right|_{y_n} \right)^{-1} G(y_n). \tag{14}$$

  Then the method is not fully implicit any more, but it may be quite stable.
  This simplified method corresponds to linearizing the nonlinearity around the current solution.
  **Example:** CN for $\frac{dy}{dt} = f(y)$

$$\begin{aligned} y_{n+1} - y_n &= \frac{1}{2}\Delta t \left(F((y_{n+1}) + F(y_n)\right) \\ &= \frac{1}{2}\Delta t \left(F(y_n + (y_{n+1} - y_n)) + F(y_n)\right) \\ &\approx \frac{1}{2}\Delta t \left( F(y_n) + \left. \frac{dF}{dy} \right|_{y_n} (y_{n+1} - y_n) + F(y_n) \right) \end{aligned}$$

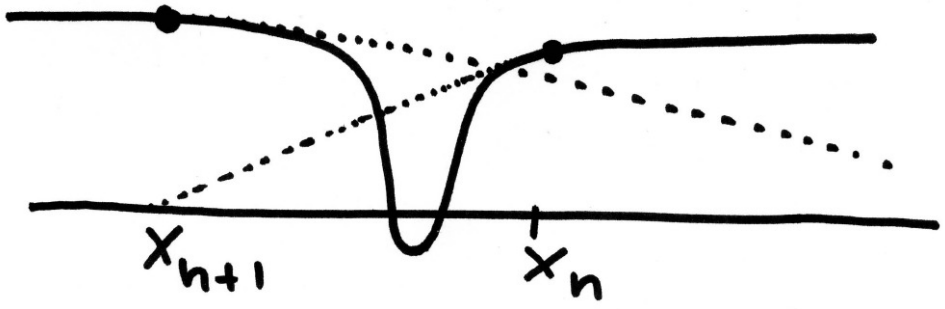
  Solve for $y_{n+1}$

$$y_{n+1} = y_n + \frac{\Delta t}{1 - \frac{1}{2}\Delta t \left. \frac{dF}{dy} \right|_{y_n}} F(y_n)$$

  Compare with the general expression (14) using (12):

$$G(y_n) = y_n + \frac{1}{2}\Delta t \left( F(u)|_{y_n} + F(y_n) \right) - u|_{y_n} = \Delta t\, F(y_n) \qquad \text{and} \qquad \left. \frac{dG}{du} \right|_{y_n} = \frac{1}{2}\Delta t \left. \frac{dF}{du} \right|_{y_n} - 1$$

**Difficulty:**
The regular Newton iteration may not converge *at all,* when initial guess is not close enough to the root: the extrapolation may completely *overshoot* the root.



One can introduce a relaxation parameter $\omega$

$$u^{(l+1)} = u^{(l)} - \omega \left( \left. \frac{dG(u)}{du} \right|_{u^{(l)}} \right)^{-1} G(u^{(l)}) \qquad 0 < \omega \leq 1 \tag{15}$$

Choice of $\omega$:

- 'downhill' method:
  insist that $|G(u^{(l)})|$ decreases under the iteration:
  since $G$ decreases *locally* in the direction defined by the Jacobian decrease $\omega$ by factors of 2 until $|G(u^{(l+1)})| < |G(u^{(l)})|$
- step-size limitation:
  - extrapolation with the slope (Jacobian) is based on *local* information, for large steps higher-order corrections can become significant
    $\Rightarrow$ if $|u^{(l+1)} - u^{(l)}|$ is large decrease $\omega$, e.g.

    $$\omega = \frac{1}{1 + \epsilon \ln(1 + \Delta^2)} \qquad \text{with} \qquad \Delta = \left| \left( \left. \frac{dG(u)}{du} \right|_{u^{(l)}} \right)^{-1} G(u^{(l)}) \right|$$

    in order for the size of the step taken to increase with $\Delta$ the factor $\omega$ should decrease more slowly than $\Delta^{-1}$with $\Delta$.
  - another strategy is a 'soft' version of the 'downhill' method:
    * decrease the step size if $G(u^{(l)}) > G(u^{(l-1)})$,

      $$\omega \to \frac{1}{2}\omega,$$

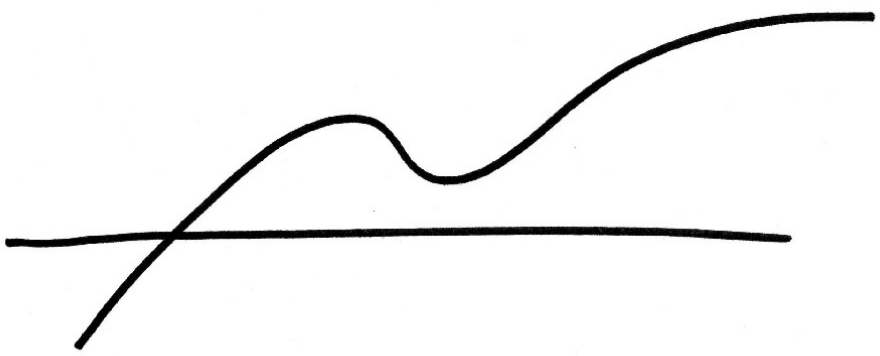
    * and increase it if $G(u^{(l)}) \leq G(u^{(l-1)})$,

      $$\omega \to \min(1, 1.1\omega).$$

    * The adjustment could also be based on a comparison of $G(u^{(l+1)})$ and $G(u^{(l)})$.

**Notes:**

- in both methods $\omega \to 1$ as the root is approached: *quadratic* convergence of Newton method is preserved asymptotically
- downhill method is very robust, cannot diverge, i.e. $y$ is always bounded
  **but:** if the root is on the other side of a maximum it cannot climb that hill, must 'tunnel' through it
- step-size limitation can climb hills
  **but:** it can diverge, depends on an arbitrary parameter $\epsilon$ that may have to be tuned.
  also the functional form $\omega(\Delta)$ is chosen heuristically.



**Note:**

- More robust for finding zeros of a function is *bisection.* It is, however, much slower: in each step error goes down only by a *constant* factor of 2, i.e the convergence is only exponential.

## 6.1 Approximate Newton Iteration

Implicit schemes require in general solving nonlinear equation for $y_{n+1}$: Newton's method

For $N$ coupled equations Jacobian is a $N \times N$-matrix:

- for large $N$ Jacobian **J** is expensive to invert
- **J** may have to be determined numerically

Perform *approximate* Newton iteration

General implicit scheme can be written in the form

$$\mathbf{y}_{n+1} = \mathbf{b} + \Delta t \mathbf{F}(\mathbf{y}_{n+1}) \tag{16}$$

where $F$ and $b$ depend also on $y_m$ with $m \leq n$.

Thus

$$\mathbf{G}(\mathbf{u}) = 0 \qquad \text{with} \qquad \mathbf{G}(\mathbf{u}) = \mathbf{b} + \Delta t \mathbf{F}(\mathbf{u}) - \mathbf{u} \qquad \text{and} \qquad \mathbf{u} = \mathbf{y}_{n+1}$$

Full Newton is given by

$$u_j^{(l+1)} = u_j^{(l)} - \sum_k (J^{-1})_{jk} \, G_k(\mathbf{u}^{(l)}) \qquad \text{with} \qquad J_{jk} = \left. \frac{\partial G_j}{\partial u_k} \right|_{\mathbf{u}^{(l)}}$$

Replace Jacobian by an approximation $\hat{\mathbf{J}}$ to the Jacobian

$$u_j^{(l+1)} = u_j^{(l)} - \sum_k (\hat{J}^{-1})_{jk} G_k(u^{(l)})$$

Will the iteration still converge?
In the vicinity of the zero $\mathbf{u}^\infty$ of $\mathbf{G}$ the approximate iteration can be approximated by expanding $u_j^{(l)} = u_j^\infty + \delta u_j^{(l)}$,

$$\begin{aligned} \delta u_j^{(l+1)} &= \delta u_j^{(l)} - \sum_k (\hat{J}^{-1})_{jk} \sum_m J_{km}(u^\infty) \delta u_m^{(l)} = \sum_m M_{jm} \delta u_m^{(l)} \\ \text{with} \qquad M_{jm} &= \delta_{jm} - \sum_k (\hat{\mathbf{J}}^{-1})_{jk} J_{km}(u^\infty) \end{aligned}$$

Such a *fixed-point iteration* converges if

$$||M_{jk}|| < 1$$

where

$$||\sum_m M_{jm} \delta u_m|| \;\leq\; ||M||\,||\delta u|| \qquad \text{for all vectors } \delta u$$

since then $||\delta u^{(l+1)}|| < ||\delta u^{(l)}||$ for all $l$.

Thus need for convergence that $\hat{\mathbf{J}}$ is close to $\mathbf{J}$

$$||\hat{\mathbf{J}} - \mathbf{J}|| < ||\hat{\mathbf{J}}||$$

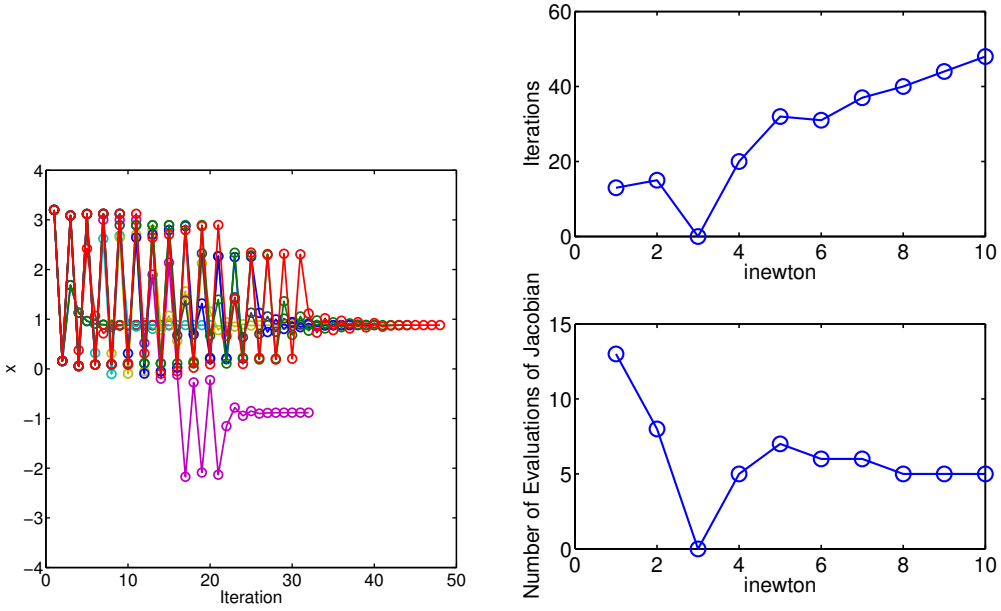
**Strategy:**

- Calculate $\mathbf{J}$ and its inverse not in each iteration but keep the same $\mathbf{J}$ for a number of iteration steps $\Rightarrow$ much faster iteration
- Convergence will be not as good
  - takes more steps
  - domain of attraction may be smaller

**Example:**

$$G(x) = a - \frac{1}{\cosh(x)}$$

For $a = 0.5$, $x_0 = 3.2$ one gets




Skipping to update the Jacobian every $inewton - 1$ steps can make the Newton fail to converge (in the plot this is indicated by 0 iterations for $inewton = 3$.

**Notes:**

- When the nonlinear equation (16) is solved only approximately, the remaining error plays a role that is similar to the round-off error. For very small time steps, $\Delta t < \Delta t_{opt}$, the error increases with decreasing time step, cf. (85). When decreasing $\Delta t$, the tolerance for solving (16) may have to be decreased as well.

# 7 Backward-Difference Formulae

As implicit scheme we had so far those of Adams-Moulton type

- backward Euler = AM1
- Crank-Nicholson = AM2

We are interested in the behavior for large $\lambda \Delta t$:

Crank-Nicholson:

$$z = 1 + \frac{1}{2}\Delta t \lambda (1 + z)$$

$$z = \frac{1 + \frac{1}{2}\Delta t\lambda}{1 - \frac{1}{2}\Delta t\lambda} = \frac{\frac{2}{\Delta t\lambda} + 1}{\frac{2}{\Delta t\lambda} - 1} \to -1\left(1 - \frac{4}{\Delta t|\lambda|}...\right) \qquad \text{for } \Delta t \to \infty \text{ and } \lambda < 0 \text{ real}$$

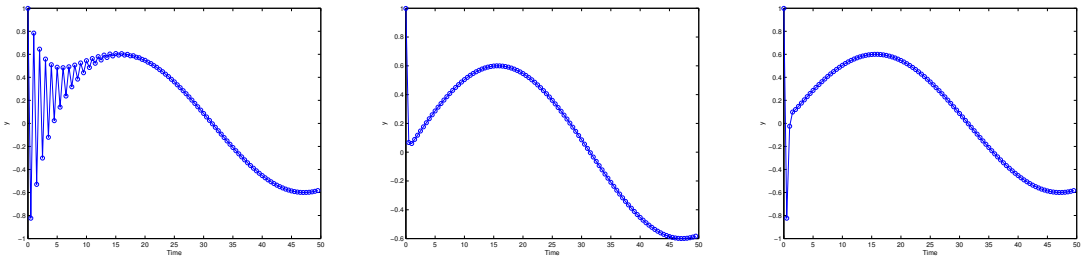


Figure 11: Different degrees of damping of the fast modes. a) Crank-Nicholson exhibits fast oscillations, b) Backward Euler damps rapidly without oscillations. c) Backward-Differencing damps with a single overshoot. $dy/dt = \lambda y + A \sin \omega t$, $\lambda = -50$, $\omega = 0.1$, $A = 30$, $\Delta t = 0.5$.

Backward Euler

$$z = \frac{1}{1 - \Delta t \lambda} \to \frac{1}{\Delta t |\lambda|} \qquad \text{for } \Delta t \to \infty \text{ and } \lambda < 0 \text{ real}$$

**Thus:**

Fast modes have large values of $\Delta t \lambda$, when they are not resolved.

- Crank-Nicholson does not capture the rapid decay of the fast mode; instead it exhibits oscillations that are only slowly damped. The period of the oscillations is $2\Delta t$, which clearly shows that they are a numerical artifact.
- Backward Euler provides strong damping of these modes. Even if they are not resolved they decay away rapidly.

**But**: Backward Euler is only a $1^{st}-$order scheme.

**Note:**

- Using a code with *large* $\Delta t \lambda$ is only meaningful if the corresponding fast modes are not *active*, i.e. if they have already decayed.
- If the fast modes are active and their fast evolution is relevant then $\Delta t \lambda$ always has to be small.

**Goal:** Develop a higher-order scheme *with strong damping*

Adams-Moulton schemes are based on *quadrature*:
We interpolated $F(t, y(t))$ over the interval $[t_{n+1-l}, t_{n+1}]$ and use the interpolating polynomial to evaluate the integral $\int_{t_n}^{t_{n+1}} F(y(t), t)\, dt$.

**Now:**
Interpolate $y$ and obtain from the polynomial the *derivative* $\frac{dy}{dt}\big|_{t_{n+1}}$ at the *new time step* $t_{n+1}$

$$\left.\frac{dy}{dt}\right|_{t_{n+1}} = F(t_{n+1}, y_{n+1}).$$

Use again Lagrange polynomials

$$p_N(t) = \sum_{k=0}^{N-1} y(t_{n+1-k}) L_k^{(N)}(t)$$

with

$$L_k^{(N)}(t) = \prod_{k \neq m=0}^{N-1} \frac{t - t_{n+1-m}}{t_{n+1-k} - t_{n+1-m}}$$

Recall

$$L_k^{(N)}(t_{n+1-l}) = \delta_{kl}.$$

1. $N = 2$

$$\begin{aligned} p_2(t) &= y(t_{n+1}) L_0^{(2)}(t) + y(t_n) L_1^{(2)}(t) \\ &= y_{n+1} \frac{t - t_n}{t_{n+1} - t_n} + y_n \frac{t - t_{n+1}}{t_n - t_{n+1}} \\ &= \frac{y_{n+1} - y_n}{t_{n+1} - t_n} (t - t_n) + y_n \end{aligned}$$

Approximate the derivative

$$\left.\frac{dy}{dt}\right|_{t_{n+1}} \approx \frac{d}{dt} p_2(t) = \frac{y_{n+1} - y_n}{t_{n+1} - t_n}$$

Thus

$$y_{n+1} = y_n + \Delta t\, F(t_{n+1}, y_{n+1}) + \mathcal{O}(\Delta t^2) \qquad \text{backward Euler}$$

2. $N = 3$

$$\begin{aligned} p_3(t) &= \frac{(t - t_{n-1})(t - t_n)}{(t_{n+1} - t_{n-1})(t_{n+1} - t_n)} y_{n+1} + \frac{(t - t_{n-1})(t - t_{n+1})}{(t_n - t_{n-1})(t_n - t_{n+1})} y_n + \frac{(t - t_n)(t - t_{n+1})}{(t_{n-1} - t_n)(t_{n-1} - t_{n+1})} y_{n-1} \\ &= \frac{1}{\Delta t^2} \left( \frac{1}{2}(t - t_{n-1})(t - t_n) y_{n+1} - (t - t_{n-1})(t - t_{n+1}) y_n + \frac{1}{2}(t - t_n)(t - t_{n+1}) y_{n-1} \right) \end{aligned}$$

$$\frac{d}{dt} p_3(t) = \frac{1}{\Delta t^2} \left( \frac{1}{2} y_{n+1}(t - t_n + t - t_{n-1}) - (t - t_{n+1} + t - t_{n-1}) y_n + \frac{1}{2}(t - t_{n+1} + t - t_n) y_{n-1} \right)$$

$$\frac{d}{dt} p_3(t_{n+1}) = \frac{1}{\Delta t} \left( \frac{3}{2} y_{n+1} - 2 y_n + \frac{1}{2} y_{n-1} \right)$$

Thus

$$\frac{3}{2} y_{n+1} - 2 y_n + \frac{1}{2} y_{n-1} = \Delta t\, F(t_{n+1}, y_{n+1}) + \mathcal{O}(\Delta t^3)$$

For variable time steps with $\Delta t_n = t_{n+1} - t_n$ one gets

$$\frac{d}{dt} p_3(t_{n+1}) = \frac{2\Delta t_n + \Delta t_{n-1}}{\Delta t_n (\Delta t_{n-1} + \Delta t_n)} y_{n+1} - \frac{\Delta t_{n-1} + \Delta t_n}{\Delta t_{n-1} \Delta t_n} y_n + \frac{\Delta t_n}{\Delta t_{n-1} (\Delta t_{n-1} + \Delta t_n)} y_{n-1}$$

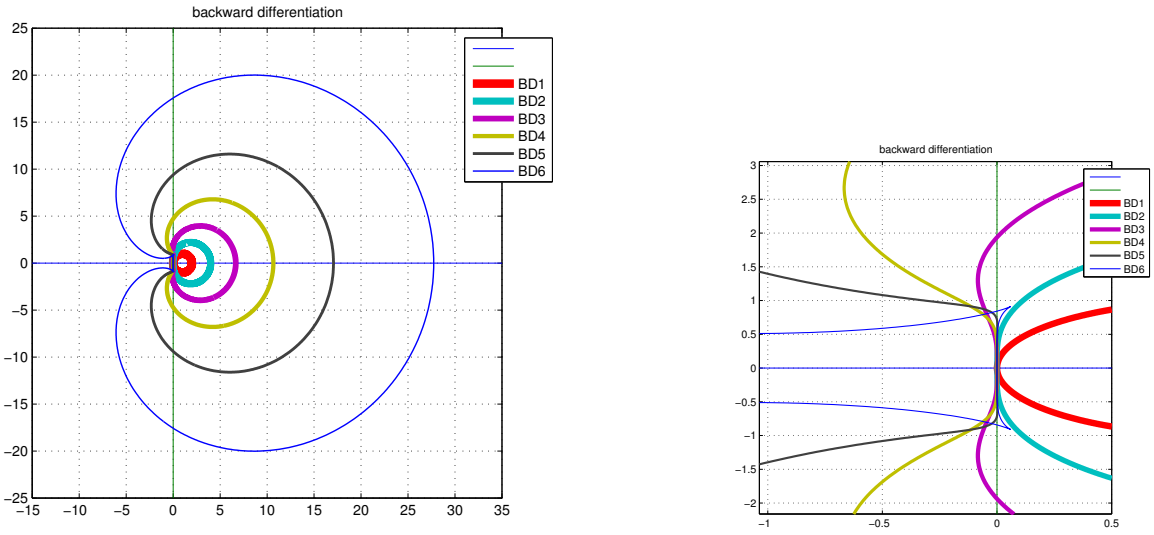



Figure 12: a) Regions of absolute stability of Backward Differencing Schemes. b) Enlargement of a). The regions shrink with increasing order.

and the numerical scheme can be written as

$$\frac{2\Delta t_n + \Delta t_{n-1}}{\Delta t_{n-1} + \Delta t_n} y_{n+1} - \frac{\Delta t_{n-1} + \Delta t_n}{\Delta t_{n-1}} y_n + \frac{\Delta t_n}{\Delta t_{n-1}} \frac{\Delta t_n}{\Delta t_{n-1} + \Delta t_n} y_{n-1} = \Delta t_n F(t_{n+1}, y_{n+1}) + \mathcal{O}(\Delta t^3)$$

**Stability**:

Stability analysis of the backward differencing scheme of $2^{nd}-$order:

$$\frac{3}{2}z - 2 + \frac{1}{2z} - \Delta t \lambda z = 0$$

$$z_{1,2} = \frac{2 \pm \sqrt{1 + 2\Delta t \lambda}}{3 - 2\Delta t \lambda}$$

For small $\Delta t \lambda$

$$z_1 \to 1 \qquad \text{scheme is consistent}$$
$$z_2 \to \frac{1}{3}$$

**Notes:**

- Multi-step schemes that use values from at least 3 different values of $t$ (BD2, AB2, AM3, ...) have multiple values of $z$:
  - Only one of the modes corresponds to a physical mode. For the scheme to be consistent that mode has to satisfy $z \to 1$ for $\Delta t \to 0$
  - All the additional modes are *unphysical*.

- The unphysical modes could lead to additional instabilities and could have $|z| > 1$ even for $\Delta t \to 0$.
- For stability we need that *all* modes are stable, i.e. we need for *all* $z$: $|z_i| \leq 1$.
- Multistep schemes like BD2 require also the solution at $t_{n-1}$. Therefore one cannot use them for the first time step for which only $y_n$ is available. For that time step use a one-step method. Since that scheme is only used for a single time step it can be one order less than the multistep scheme used. For a BD2 scheme one could simply use backward Euler.

For large $\Delta t\lambda$

$$z_{1,2} = \frac{2 \pm \sqrt{1 + 2\Delta t\lambda}}{3 - 2\Delta t\lambda} \to \pm i \frac{1}{\sqrt{2\Delta t\,|\lambda|}} \to 0 \qquad \text{for } \Delta t \to \infty \text{ and } \lambda < 0 \text{ real}$$

**Note:**

- BD1 and BD2 are *unconditionally stable* (stable *outside* the closed region in Fig.12)
- Comparing computational effort: in Crank-Nicholson the rhs has to be evaluated twice whereas for BD2 only once.
- BD3, BD4, BD5, and BD6 have *unstable* regions
- BD2 damps fast modes *and* is $2^{nd}-$order. In contrast to backward Euler, however, even for $\lambda$ real $z$ becomes imaginary for $\Delta t > \frac{1}{2|\lambda|}$, which implies an oscillatory response even though the differential equation itself exhibits no oscillations. Thus, the fast modes decay rapidly, but they do so in an oscillatory fashion, which manifests itself as an overshoot.

## 7.1 Brief Comments on Backward-Euler and Backward-Difference Code

We are solving backward Euler,

$$y_{n+1} = y_n + \Delta t\, F(y_{n+1}, t_{n+1})$$

or backward-difference $2^{nd}$ order

$$\frac{3}{2} y_{n+1} - 2y_n + \frac{1}{2} y_{n-1} = \Delta t\, F(y_{n+1}, t_{n+1})$$

In each time step we have a nonlinear equation to solve, which has the form

$$G_{BE}(u) \equiv -u + y_n + \Delta t\, F(u, t_{n+1}) = 0 \qquad G_{BD2}(u) \equiv -\frac{3}{2}u + 2y_n - \frac{1}{2}y_{n-1} + \Delta t\, F(u, t_{n+1}) = 0$$

We do this using Newton iteration with a relaxation parameter $\omega$

$$u_i^{l+1} = u_i^l - \omega \left( \frac{\partial G_i(u^l)}{\partial u_j} \right)^{-1} G_j(u^l).$$

Determine the Jacobian J numerically

$$J_{ij} \equiv \frac{\partial G_i(u^l)}{\partial u_j} \approx \frac{G_i(u_1^l, ..., u_j^l + \delta u_j, ...u_N^l) - G_i(u_1^l, ..., u_j^l, ...u_N^l)}{\delta u_j}$$

with $|\delta u_j| \ll 1$ (code is on the class web site).

Thus, we need to define the function *G(u)* , which has a variable that picks which scheme is to be done. That function is called by the function *dG* and by the function *step_implicit,* which performs a single time step, implementing the Newton iteration.

In the Newton iteration we include the relaxation parameter $\omega$

$$\omega^{(l)} = \frac{1}{1 + \delta \log\left(1 + (\Delta^{(l)})^2\right)} \qquad \Delta^{(l)} = \left\| \frac{d\tilde{u}_i}{\max(Y_i(t_n), 10^{-10})} \right\|$$

with $\Delta^{(l)}$ measuring the relative change in $u$

$$d\tilde{u}_i = -\sum_{jk} \left(\mathbf{J}^{-1}\right)_{ij} G_k(\mathbf{u}^{(l)})$$

hat would be taken in a full Newton step.

But actually we solve this in the form

$$\mathbf{J}\left(\mathbf{u}^{(l+1)} - \mathbf{u}^{(l)}\right) = -\omega \mathbf{G}\left(\mathbf{u}^{(l)}\right) \qquad \text{i.e. in matlab} \quad du = omega * J \backslash G(u)$$

Do not calculate the inverse Jacobian, but use Matlab's backslash operator to determine $u^{l+1}$, which computes $u^{l+1}$ by solving the equation multiplied by the Jacobian:

$$\mathbf{x} = \mathbf{A}^{-1}\mathbf{b} \qquad \Rightarrow \qquad \mathbf{A}\mathbf{x} = \mathbf{b} \qquad \Rightarrow \qquad \mathbf{x} = \mathbf{A} \backslash \mathbf{b}$$

The convergence of the Newton can be measured by $\Delta u = abs(u^{l+1} - u^l)$. If the Newton converges, it converges very rapidly and one can usually pick the tolerance for $\Delta u$ very small ($\mathcal{O}(10^{-10})$). To avoid effectively endless loops when the Newton does not want to converge limit the number of Newton steps to some number $inewtonmax$, i.e. break out of the Newton if the limit is reached and stop the computation.

In each time step perform adaptive step-size control as before by comparing the result obtained using a single step $\Delta t$ with that obtained with two steps of size $\Delta t/2$.

**Note:**

- for BD2 one needs $y_{n-1}$ and would, in principle, have to use a different scheme (e.g. BE) for the very first time step. If the initial transients are not essential, one can sometimes simply set

$$y_2 = y_1 = y(t = 0).$$

This assumes that during the first time step the parameters of the equation (i.e. $I_{inj}$) are assumed to be chosen such that the solution has a steady state and that the initial condition corresponds to that steady state.

# 8 Chemical Oscillations in the Belousov-Zhabotinsky System

In the 1950s Belousov was investigating catalysis in the Krebs cycle, which underlies the energy production in mitochondria in eukaryotic cells. In a modified version of the reaction he found persistent oscillations in the concentrations of the reactants. He could not get his results published because they 'contradicted the laws of physics' [12]:

- The response from the editor of the first journal was: the "supposedly discovered discovery" is quite impossible. The manuscript can only be published if existing theories can be shown to be flawed.
- After 6 years of more careful experiments to decipher the mechanism he submitted an extended study to another journal. He was told that it can only be published as a claim, with the manuscript truncated like a letter to the editor without the new convincing evidence.

Belousov did not publish it at all. His recipe circulated among faculty. In the early 60s, Zhabotinsky as a graduate student followed a suggestion of his adviser Schnoll experimented with a recipe of unknown origin and told Belousov about his results. Belousov sent him an unpublished manuscript which Zhabotinsky cited in his paper. But they never met. Belousov, Zhabotinsky, Krinsky and Ivanitsky received the Lenin Prize in 1980. Belousov had died, however, already in 1970.

The BZ reaction has been relevant in particular from a fundamental point of view as showing that far from thermodynamic equilibrium chemical systems can spontaneously form temporal structures, even chaotic dynamics[13]. In Fig.13, note the very steep temporal gradients suggesting stiff dynamics.

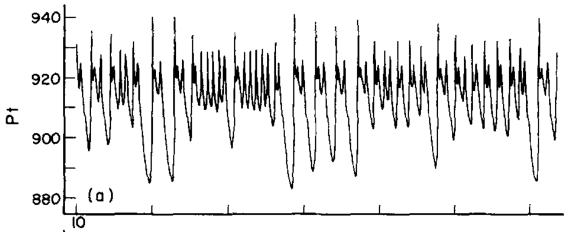



Figure 13: Chaotic oscillations in the Belousov-Zhabotinsky reaction.

Chemical oscillations functionally important in many biological systems (e.g. Ca-waves in eggs upon fertilication, circadian clock, hormonal cycles, ovarian cycle ).

[12] cf. A.T. Winfree, *The Prehistory of the Belousov-Zhabotinsky Oscillator*, J. Chemical Education 61 (1984) 661.

[13] R.A. Schmitz, K.R. Graziani, and J.L. Hudson, J. Chem. Phys. 67 (1977) 3020.

A very interesting function of chemical oscillations is in the communication between organisms with chemical waves between organisms:

Individual cells of dictyostelium discoidum (Slime Mold) use cAMP waves to organize themselves and aggregate[14]. Starvation of colonies of those cells trigger the following steps (see Figs.14,15)

1. Individual slime mold cells start emitting cAMP waves; other cells respond to that cAMP and relay the cAMP waves. This sets up concentric waves or spiral waves.
2. The slime mold cells then start streaming along the gradients of the cAMP concentration and coalesce to form a mound.
3. The mound forms a stalk and a fruiting body, which disperses then the cells over larger distances to seek 'greener pastures'.

Videos of BZ:

- http://www.youtube.com/watch?v=lOocwfCE5Cs
- good documentary with history:
  http://www.youtube.com/watch?v=nEncoHs6ads . It shows a graph of the temporal evolution of colors at min 10:20
- Spirals induced by cutting waves:
  http://www.youtube.com/watch?v=3JAqrRnKFHo

Videos of Dictyostelium discoidum evolution

- spiral waves and onset of streaming
  http://www.youtube.com/watch?v=OX5Yiz38fgY
- long duration of slime mold moving (slug)
  http://www.youtube.com/watch?v=uqi_WTllG7A
- aggregation and formation of fruiting body
  http://www.youtube.com/watch?v=tpdIvlSochk

A classical model for the BZ reaction is that developed by Field and Noyesin the 1970's[15], which they called the 'Oregonator' (they were at the U. Oregon at that point; there had been another simpler 3-component model for chemical oscillations proposed by Prigogine et al. in Brussels, which was called 'Brusselator' by Tyson).

[14]for a review see F. Siegert and C.J. Weijer, *Spiral and concentric waves organize multicellular Dicyostelium mounds,* Current Biology 5 (1995) 937. The figures are from that paper.

[15]R.J. Field and R.M. Noyes, J. Am. Chem. Soc. 96 (1974) 2001.

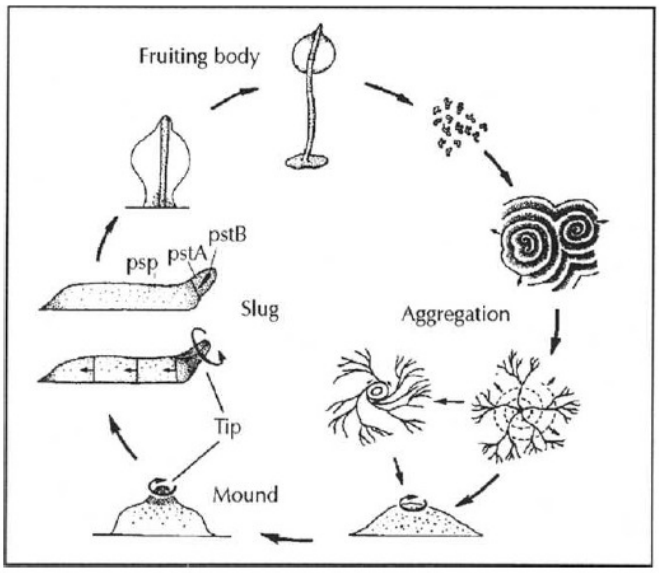



**Fig. 1.** Life-cycle of the cellular slime mould *Dictyostelium discoideum*. Upon starvation, single cells begin to move chemotactically towards an aggregation centre. The chemotactic signal is visible as an expanding optical-density spiral originating from the aggregation centre. After about 20 waves, cells collect in bifurcating aggregation streams and finally form a hemispherical cell mass, the mound. In the mound, the cells differentiate at random into at least three types: prespore cells (psp) and two prestalk cell types (pstA, pstB). The prestalk cells sort chemotactically to the top of the mound and form the tip. In the slug tip, the pstB cells form a funnel-shaped structure, from which the stalk develops. The pstA cells surround the pstB cells. The tip directs morphogenesis until fruiting-body formation. Circular arrows indicate the direction of wave propagation.

Figure 14: Dictyostelium discoidum aggregation cycle.

It has three components describing Bromous acid, Bromide, and Cerium concentrations. An extension to include the photosensitivity of the catalyst $Ru(bpy)_3^{3+}$ has been published by Amemiya, Ohmori, and Yamaguchi. It is given by

$$\begin{pmatrix} \dot{x} \\ \dot{y} \\ \dot{z} \\ \dot{v} \end{pmatrix} = \begin{pmatrix} k_{01}Ay - k_{02}xy + k_{03}Ax - 2k_{04}x^2 \\ -k_{01}Ay - k_{02}xy + hk_{05}z \\ 2k_{03}Ax - k_{05}z \\ k_{01}Ay + 2k_{02}xy + k_{04}x^2 - k_{06}v \end{pmatrix} + \begin{pmatrix} -x \\ -(y - y_{in}) \\ -z \\ -(v - v_{in}) \end{pmatrix} k_f + \begin{pmatrix} p_2 \\ p_1 \\ p_1 + 2p_2 \\ -p_1 \end{pmatrix} \phi \tag{17}$$

Here the variables are

$$\begin{aligned} x &= [HBrO_2] \quad \text{Bromous Acid} \\ y &= [Br^-] \quad \text{Bromide} \\ z &= [Ru(bpy)_3^{3+}] \quad \text{dipyridine Ruthenium} \\ v &= [BrMA] \quad \text{Bromomalonic acid} \\ A &= [BrO_3^-] \quad \text{Bromate} \end{aligned}$$

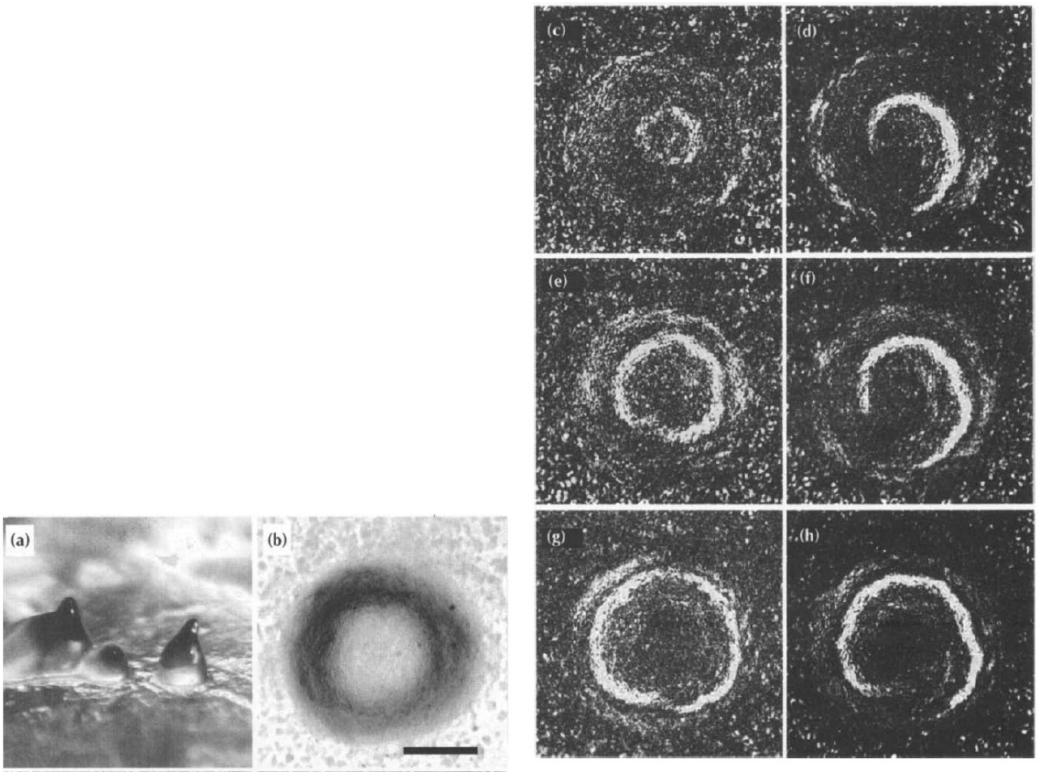



Figure 15: a) Dictyostelium discoidum mounds. b) Concentric and spiral waves.

Without going into details of the chemistry, just a couple of examples of the reactions modeled

$$A + Y \overset{k_{01}}{\rightarrow} X + V \tag{18}$$

$$X + Y \overset{k_{02}}{\rightarrow} 2V \qquad \text{removal of X via feedback through Y and Z} \tag{19}$$

$$A + X \overset{k_{03}}{\rightarrow} 2X + 2Z \qquad \text{autocatalytic step} \tag{20}$$

$$2X \overset{k_{04}}{\rightarrow} V + A \tag{21}$$

$$Z \overset{k_{05}}{\rightarrow} hY \tag{22}$$

One obtains the differential equations using the law of mass action. It can be visualized by imagining that reactions occur when the reactants collide with each other. Assuming that they are being mixed randomly the probability of a collision is proportional to the concentrations of all participating reactants. For instance, in reaction (21) two $HBrO_2$ molecules have to collide ($x^2$), which then produces one $BrMA$ and one $BrO_3^-$ molecule each. Two $HBrO_2$ molecules disappear for each reaction. Some reactants are assumed to be present in much higher concentrations, which do not change appreciably during the concentration; they are given by constants in (17).

The essential ingredients for the oscillations are the autocatalytic production of $X$ in (20), which also generates $Z$, which then forms $Y$ (22), which in turn removes $X$ in (19) and therefore forms a delayed negative feedback.

In addition, there is influx of $y$ and $v$ from a reservoir with concentrations $y_{in}$ and $v_{in}$, respectively. A realization can be diffusion across a membrane that separates the reservoir from the reaction compartment.

The light-induced reactions are captured with $p_{1,2}$ with $\phi$ being the light flux.

Like many chemical reactions, the system is quite stiff: the reaction rates vary substantially

$$k_{01} = 2H^2 \frac{1}{M^3 s} \qquad k_{02} = 3 \cdot 10^6 H \frac{1}{M^2 s}$$
$$k_{03} = 42H \frac{1}{M^2 s} \qquad k_{04} = 3 \cdot 10^3 \frac{1}{Ms} \qquad k_{06} = 2 \cdot 10^{-3} \frac{1}{s}$$

where $H = [H^+]$ is of $\mathcal{O}(1M)$.

In their paper Amemiya et al. they show that the model can reproduce the light dependence of the reaction and generate complex dynamics, aspects of which are also seen in experiments.

# 9 Multi-Step Methods

Quadrature approach

$$y_{n+1} = y_n + \int_{t_n}^{t_{n+1}} F(t, y(t'))\, dt'.$$

Effectively, the integrand was approximated by a polynomial.

Single-step method used only values between $t_n$ and $t_{n+1}$ to obtain the polynomial

Multi-step methods use also previous time steps $t_{n+1}$, $t_n$, $t_{n-1}$, $t_{n-2}$, $t_{n-3}$,... as input for the polynomial:

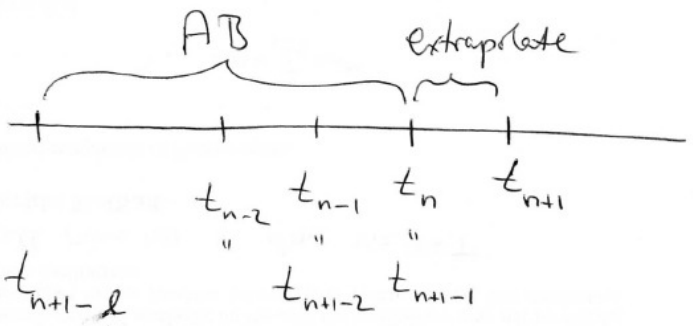
- *explicit*: do not include $t_{n+1}$, the integration involves an extrapolation to $[t_{n+1}, t_n]$
- *implicit*: include $t_{n+1}$, the integration uses interpolation within $[t_{n+1}, t_n]$

## 9.1 Adams-Bashforth Methods[16]

Explicit method:

Use a polynomial $P_N(t)$ that interpolates $F(t, y(t))$ on the 'grid points' $t_l$ in the range $[t_{n+1-N}, t_n]$, $N \geq 1$, at which $y$ and with it $F$ is known ,

$$P_N(t_l) = F(t_l, y(t_l))$$




Look for a polynomial $L_k^{(N)}(t)$ that satisfies

$$L_k^{(N)}(t_{n+1-l}) = \delta_{k,l}$$

[16] John Couch Adams (1819-1892), predicted the existence of Neptune based on perturbation calculations (independently also predicted by U. Le Verrier).

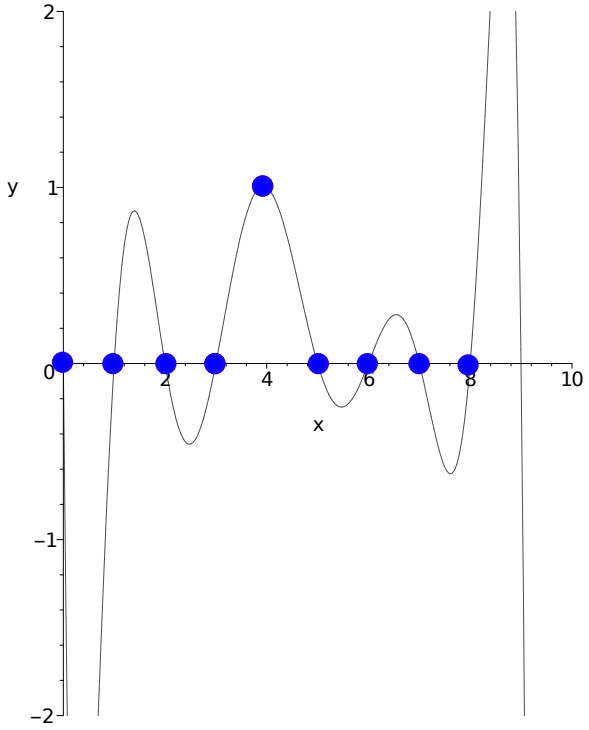



Figure 16: Lagrange polynomial $L_4^{(N)}(t)$ vanishes on all grid points except for $t_{n+1-4}$.

Why would that be useful? Define the function

$$P_N(t) = \sum_{k=1}^{N} F(t_{n+1-k}, y_{n+1-k}) L_k^{(N)}(t)$$

It satisfies then

$$\begin{aligned} P_N(t_{n+1-l}) &= \sum_{k=1}^{N} F(t_{n+1-k}, y_{n+1-k}) L_k^{(N)}(t_{n+1-l}) \\ &= \sum_{k=1}^{N} F(t_{n+1-k}, y_{n+1-k}) \delta_{k,l} \\ &= F(t_{n+1-l}, y_{n+1-l}) \end{aligned}$$

Thus, $P_N(t)$ interpolates $F(t, y)$ on the temporal grid $t_n$.

Use polynomials for $L_k^{(N)}(t)$. Any polynomial can be written as

$$(t - \alpha_1)(t - \alpha_2)...(t - \alpha_M)$$

The polynomial vanishes at all $t = \alpha_l$, choose the $\alpha_l$ at the 'grid points'

$$\alpha_l = t_{n+1-l}$$

The polynomial should not vanish at $t_{n+1-k}$: skip the factor involving $\alpha_k = t_{n+1-k}$

$$(t - t_n)(t - t_{n-1})...(t - t_{n+1-(k-1)}) \cdot \ 1 \ \cdot (t - t_{n+1-(k+1)})...(t - t_{n+1-N})$$

We need to normalize the polynomial still at $t = t_{n+1-k}$: introduce Lagrange polynomials $L_k^{(N)}(t)$ of order $N-1$

$$\begin{aligned} L_k^{(N)}(t) &= \frac{t-t_n}{t_{n+1-k}-t_n}\frac{t-t_{n-1}}{t_{n+1-k}-t_{n-1}}\ldots\frac{t-t_{n+1-(k-1)}}{t_{n+1-k}-t_{n+1-(k-1)}}\cdot 1\cdot\frac{t-t_{n+1-(k+1)}}{t_{n+1-k}-t_{n+1-(k+1)}}\ldots\frac{t-t_{n+1-N}}{t_{n+1-k}-t_{n+1-N}} \\ &= \prod_{k\neq m=1}^{N}\frac{t-t_{n+1-m}}{t_{n+1-k}-t_{n+1-m}} \qquad 1\leq k\ \leq N \end{aligned}$$

On the grid points $t_{n+1-l}$

$$L_k^{(N)}(t_{n+1-l}) = \prod_{k\neq m=1}^{N}\frac{t_{n+1-l}-t_{n+1-m}}{t_{n+1-k}-t_{n+1-m}}$$

- for $l \neq k$ in the product there is one term with $m = l$: numerator vanishes:

$$L_k^{(N)}(t_{n+1-l}) = 0$$

- for $l = k$ the value $m = l$ is skipped in the product and in all terms the numerator and denominator are equal:

$$L_k^{(N)}(t_{n+1-l}) = 1$$

Thus

$$L_k^{(N)}(t_{n+1-l}) = \delta_{k,l}$$

and we have the interpolant

$$P_N(t) = \sum_{k=1}^{N} F(t_{n+1-k}, y_{n+1-k})L_k^{(N)}(t)$$

**Examples:**

1. $N = 1$

$$L_1^{(1)}(t) = 1 \qquad P_1(t) = F(t_n, y_n)\cdot 1$$

$F$ is approximated by a constant.

2. $N = 2$

$$L_1^{(2)}(t) = \frac{t-t_{n-1}}{t_n-t_{n-1}}, \qquad L_2^{(2)}(t) = \frac{t-t_n}{t_{n-1}-t_n}$$

$$\begin{aligned} P_2(t) &= F(t_n, y_n)\frac{t-t_{n-1}}{t_n-t_{n-1}} + F(t_{n-1}, y_{n-1})\frac{t-t_n}{t_{n-1}-t_n} \\ &= \frac{F(t_n, y_n)-F(t_{n-1}, y_{n-1})}{t_n-t_{n-1}}(t-t_{n-1}) + F(t_{n-1}, y_{n-1}) \end{aligned}$$

$F$ is approximated by a straight line.

**Notes:**

- One could obtain $P_N(t)$ also without using the Lagrange polynomials $L_k^{(N)}(t)$, but then one would have to solve an $N \times N$ matrix problem to get the coefficients.

The quadrature formula becomes

$$\begin{aligned} y_{n+1} &= y_n + \int_{t_n}^{t_{n+1}} \sum_{k=1}^{N} F(t_{n+1-k}, y_{n+1-k}) L_k^{(N)}(t) dt \\ &= y_n + \sum_{k=1}^{N} F(t_{n+1-k}, y_{n+1-k}) \int_{t_n}^{t_{n+1}} L_k^{(N)}(t) dt \end{aligned}$$

Simplify to a equally spaced grid $t_n = t_0 + n\Delta t$

$$y_{n+1} = y_n + \sum_{k=1}^{N} F(t_{n+1-k}, y_{n+1-k}) \int_{t_n}^{t_{n+1}} \prod_{1=m\neq k}^{N} \frac{t - t_{n+1-m}}{(m-k)\Delta t}\, dt$$

**Adams-Bashforth 1:**

$N = 1$

$$y_{n+1} = y_n + F(t_n, y_n) \int_{t_n}^{t_{n+1}} 1\, dt = y_n + F(t_n, y_n)\Delta t$$

results in forward Euler.

**Adams-Bashforth 2:**

$N = 2$

$$\begin{aligned} y_{n+1} &= y_n + F(t_n, y_n) \int_{t_n}^{t_{n+1}} \frac{t - t_{n-1}}{\Delta t}\, dt + F(t_{n-1}, y_{n-1}) \int_{t_n}^{t_{n+1}} \frac{t - t_n}{-\Delta t}\, dt \\ &= y_n + \Delta t \left( \frac{3}{2} F(t_n, y_n) - \frac{1}{2} F(t_{n-1}, y_{n-1}) \right) \end{aligned}$$

With variable time steps this becomes

$$y_{n+1} = y_n + \frac{\Delta t_n}{\Delta t_{n-1}} \left\{ \left( \frac{1}{2}\Delta t_n + \Delta t_{n-1} \right) F(t_n, y_n) - \frac{1}{2}\Delta t_n\, F(t_{n-1}, y_{n-1}) \right\}.$$

**Notes:**

- To start AB2, AB3, ... we need also $y_{n-1} = y_{-1}$ and earlier points:
  use a one-step scheme for a single time step. Its order can be one less (i.e. $1^{st}$ order for AB2); since applied only for a single time step its contribution to the error is of the same order as the global error of the rest.

- Using the Lagrange polynomials one can relatively easily generate higher-order AB-methods.

Truncation error for AB2:

$$\tau_l = \tilde{y}_n + \Delta t \frac{d\tilde{y}}{dt} + \frac{1}{2}\Delta t^2 \frac{d^2\tilde{y}}{dt^2} + \mathcal{O}(\Delta t^3) - \left( y_n + \Delta t \, (\frac{3}{2} F(t_n, y_n) - \frac{1}{2} F(t_{n-1}, y_{n-1}) \right)$$

Assume that at $t_{n-1}$ and $t_n$ we are starting with the exact solution

$$y_{n-1} = \tilde{y}_{n-1} \qquad y_n = \tilde{y}_n$$

Use Taylor expansion around $(t_n, y_n)$ for $y_{n-1}$

$$\begin{aligned} F(t_{n-1}, y_{n-1}) &= F(t_n, y_n) - \Delta t \frac{\partial F}{\partial t} - (y_n - y_{n-1}) \frac{\partial F}{\partial y} + \mathcal{O}(\Delta t^2, \Delta y^2) \\ &= F(t_n, y_n) - \Delta t \frac{\partial F}{\partial t} - \left( \Delta t F(t_n, y_n) + \mathcal{O}(\Delta t^2) \right) \frac{\partial F}{\partial y} + \mathcal{O}(\Delta t^2, \Delta y^2) \end{aligned}$$

In addition,

$$\frac{d^2\tilde{y}}{dt^2} = \frac{d}{dt} F(t, \tilde{y}) = \frac{\partial F}{dt} + \frac{\partial F}{\partial y} F$$

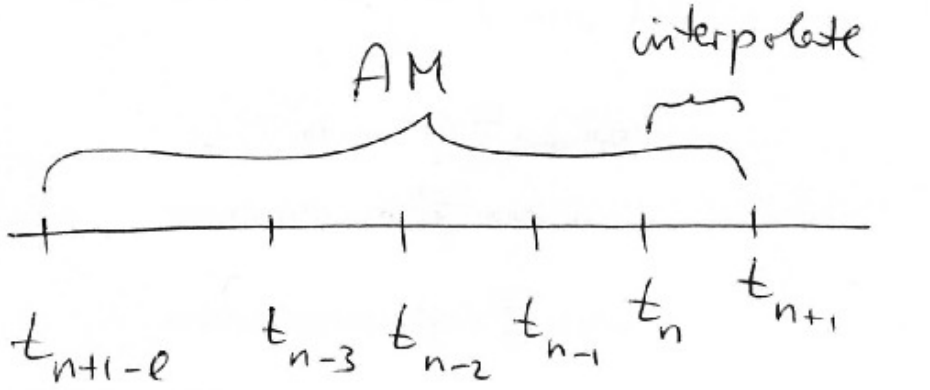
$$\begin{aligned} \tau_l &= \Delta t \left( \frac{d\tilde{y}}{dt} - (\frac{3}{2} - \frac{1}{2}) F(t_n, y_n) \right) + \Delta t^2 \left( \frac{1}{2} (\frac{\partial F}{dt} + \frac{\partial F}{\partial y} F) + \frac{1}{2} (-\frac{\partial F}{\partial t} - F \frac{\partial F}{\partial y}) \right) + \mathcal{O}(\Delta t^3) \\ &= \mathcal{O}(\Delta t^3) \end{aligned}$$

## 9.2 Adams-Moulton Methods[17]

Implicit Method:

Now include $t_{n+1}$ in the interpolation grid.

[17] Forest R. Moulton (1872-1952), astronomer.




Use again Lagrange polynomials $L_k^{(N)}(t)$ of order $N-1$

$$L_k^{(N)}(t) = \prod_{k \neq m=0}^{N-1} \frac{t - t_{n+1-m}}{t_{n+1-k} - t_{n+1-m}}$$

**Note:** The limits for the index $m$ is shifted compared to AB to include the new time $t_{n+1}$.

The interpolant is then

$$P_N(t) = \sum_{k=0}^{N-1} F(t_{n+1-k}, y_{n+1-k}) L_k^{(N)}(t)$$

For equally spaced grid $t_n = t_0 + n\Delta t$ we get

$$y_{n+1} = y_n + \sum_{k=0}^{N-1} F(t_{n+1-k}, y_{n+1-k}) \int_{t_n}^{t_{n+1}} \prod_{0=m \neq k}^{N-1} \frac{t - t_{n+1-m}}{(m-k)\Delta t}\, dt$$

**Example:**

**Adams-Moulton 1**

$$y_{n+1} \quad = \quad y_n + \Delta t F(t_{n+1}, y_{n+1})$$

results in backward Euler.

**Adams-Moulton 2:**

$$
\begin{aligned}
y_{n+1} &= y_n + F(t_{n+1}, y_{n+1}) \int_{t_n}^{t_{n+1}} \frac{t - t_n}{\Delta t}\, dt + F(t_n, y_n) \int_{t_n}^{t_{n+1}} \frac{t - t_{n+1}}{-\Delta t}\, dt \\
&= y_n + \frac{1}{2}\Delta t \left(F(t_n, y_n) + F(t_{n+1}, y_{n+1})\right)
\end{aligned}
$$

results in Crank-Nicholson scheme:

$$
\frac{y_{n+1} - y_n}{\Delta t} = \frac{1}{2}\left(F(t_n, y_n) + F(t_{n+1}, y_{n+1})\right).
$$

**Note:**

- AM2=CN can also be thought of as central difference in time for the time derivative at time $t = t_{n+1/2}$
- The main advantage of the implicit schemes is their enhanced *stability* (see below).
- In general, implicit methods require the solution of nonlinear equations at each time step $\Rightarrow$ see below.

## 9.3 Predictor-Corrector Methods

Avoid the implicit equation by using an approximation to $y_{n+1}$ on the r.h.s: use a lower-order explicit scheme to 'predict' $y_{n+1}$.

**Examples:**

1. AB1 & AM2

$$
\begin{aligned}
y^* &= y_n + \Delta t F(t_n, y_n) \\
y_{n+1} &= y_n + \frac{1}{2}\Delta t \left(F(t_n y_n) + F(t_{n+1}, y^*)\right)
\end{aligned}
$$

   This is the same scheme as the improved Euler scheme or RK2. It has a truncation error $\tau = \mathcal{O}(\Delta t^2)$.

2. AB2 & AM3

$$
\begin{aligned}
y^* &= y_n + \Delta t \left(\frac{3}{2}F(t_n, y_n) - \frac{1}{2}F(t_{n-1}, y_{n-1})\right) \\
y_{n+1} &= y_n + \Delta t \left(\frac{5}{12}F(t_{n+1}, y^*) + \frac{8}{12}F(t_n, y_n) - \frac{1}{12}F(t_{n-1}, y_{n-1})\right)
\end{aligned}
$$

Because the predicted value $y^*$ is used in the increment, which has already a factor of $\Delta t$ the predictor can always be one order less accurate than the corrector without reducing the accuracy below that of the corrector.

Show this explicitly for the special case $F(t,y) = \lambda y$

Assume that for $t_{n-1}$ and for $t_n$ we have the exact solution,

$$y_n = \tilde{y}_n \qquad y_{n-1} = \tilde{y}_{n-1},$$

and compute the truncation error that we incur in one time step

The exact solution satisfies

$$\tilde{y}_{n+1} = \tilde{y}_n + \Delta t\lambda\tilde{y}_n + \frac{1}{2}\Delta t^2\lambda^2\tilde{y}_n + \frac{1}{6}\Delta t^3\lambda^3\tilde{y}_n + \mathcal{O}(\Delta t^4).$$

Using a predicted value $y^*$ for $y_{n+1}$ AM3 is given by

$$y_{n+1} = y_n + \frac{1}{12}\Delta t\left\{5\lambda y^* + 8\lambda y_n - \lambda y_{n-1}\right\}$$

We have

$$y_{n-1} = e^{-\lambda\Delta t}y_n = y_n - \lambda\Delta t y_n + \frac{1}{2}\Delta t^2\lambda^2 y_n + \mathcal{O}(\Delta t^3).$$

Write the predicted value as

$$y^* = y_n + \alpha\Delta t + \frac{1}{2}\beta\Delta t^2 + \mathcal{O}(\Delta t^3).$$

We get then

$$\begin{aligned}\tilde{y}_{n+1} - y_{n+1} &= \tilde{y}_n - y_n + \Delta t\left\{\lambda\tilde{y}_n - \underbrace{\frac{1}{12}\left[5\lambda + 8\lambda - \lambda\right]}_{=1}y_n\right\} + \Delta t^2\left\{\frac{1}{2}\lambda^2\tilde{y}_n - \frac{5}{12}\lambda\alpha - \frac{1}{12}\lambda^2 y_n\right\} + \\ &\quad + \Delta t^3\left\{\frac{1}{6}\lambda^3\tilde{y}_n - \frac{5}{24}\beta\lambda + \frac{1}{24}\lambda^3 y_n\right\}\end{aligned}$$

Using $\tilde{y}_n = y_n$ we get the two conditions

$$\alpha = \lambda y_n \qquad \beta = \lambda^2 y_n$$

i.e. in order to obtain a local truncation error of $\mathcal{O}(\Delta t^4)$ it is sufficient if we approximate $y^*$ correctly to $\mathcal{O}(\Delta t^3)$

$$y^* = y_n + \Delta t\lambda y_n + \frac{1}{2}\Delta t^2\lambda^2 y_n + \mathcal{O}(\Delta t^3)$$

**Notes:**

- Using only the simply differential equation $\frac{dy}{dt} = \lambda y$ the calculation did not test the treatment of the term $\frac{\partial^2 F}{\partial y^2}$ that arises at $\mathcal{O}(\Delta t^3)$. But the conclusion obtained with this simpler calculation is correct.
- The predictor-corrector method is not as stable as the implicit method, but it is more stable than the AB-method of same order and it uses fewer time levels since the time-levels of of the higher-order AM-portion include $t_{n+1}$.

# 10 Application to Partial Differential Equations

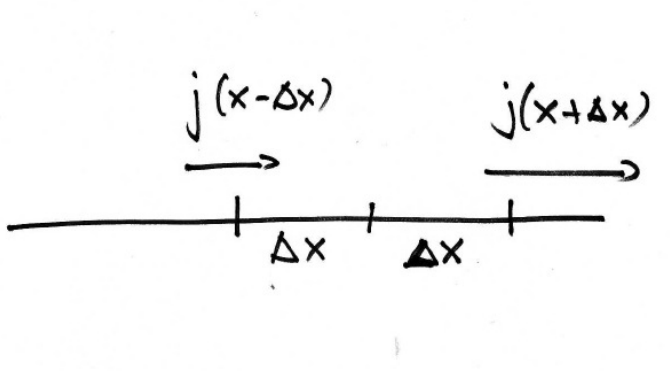
Evolution of spatially continuous systems: partial differential equations

**Examples**:

1. Fourier's Law of Heat Diffusion
   Heat flux is driven by differences in the temperature, but no curvature in the temperature profile is needed: $j = j(\partial T/\partial x)$. For small gradients one can ignore nonlinearities in the gradient and models the heat flux as being proportional to the gradient of the temperature

   $$j(x,t) = -\kappa \frac{\partial}{\partial x} T$$

   The net flux out of a small region in space of width $2\Delta x$ lasting a short time interval $\Delta t$ reduces the temperature inside that region




   $$\left(j(x+\Delta x,t) - j(x-\Delta x,t)\right)\Delta t = -2\Delta x\, c_p\ \left(T(x,t+\Delta t) - T(x,t)\right)$$

   In the limit of $\Delta t \to 0$ and $\Delta x \to 0$ the differences become derivatives,

   $$\frac{\partial j}{\partial x} = -c_p \frac{\partial}{\partial t} T.$$

   Inserting the dependence of the heat flux on the temperature gradient one obtains

   $$\frac{\partial}{\partial t} T = D \frac{\partial^2}{\partial x^2} T$$

   with

   $$D = \frac{\kappa}{c_p}$$

   Analogously: Fick's law of mass diffusion.
   To obtain a simple solution make an exponential ansatz since the differential equation has constant coefficients

   $$T(x,t) = T_0 e^{\lambda t} \cos qx$$

Inserting into the differential equation leads to the condition

$$\lambda = -Dq^2$$

Thus:

- any spatial modulation of the temperature decays over time: diffusion smooths out variations

2. Wave Equation
Analogous arguments for an elastic string where the restoring forces acting on a piece of the string enter Newton's equation of motion lead to

$$\frac{\partial^2 u}{\partial t^2} = c^2 \frac{\partial^2 u}{\partial x^2}$$

Simple solutions

(a) Plucked string that is fixed at the ends: $u(0) = 0 = u(L)$

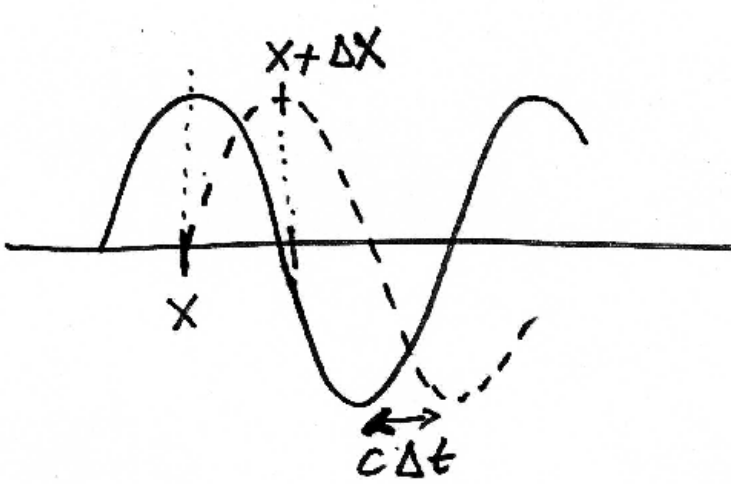
$$u(x,t) = u_0 \sin(qx)\, \cos(\omega t)$$

with $q = n\frac{\pi}{L}$. Inserting this ansatz into the wave equation yields the dispersion relation $\omega^2 = c^2 q^2$.

(b) Traveling wave

$$u(x,t) = u_0 \cos(qx - \omega t) \qquad \omega^2 = c^2 q^2$$

The maxima of the wave are traveling with a fixed speed




The maxima are at $qx - \omega t = 2\pi n$. After a short time interval $\Delta t$ they have traveled a distance $\Delta x$,

$$qx - \omega t = 2\pi n = q(x + \Delta x) - \omega(t + \Delta t) \qquad \Rightarrow \qquad q\Delta x - \omega \Delta t = 0$$

The wave speed is therefore given by

$$v_\varphi = \frac{\Delta x}{\Delta t} = \frac{\omega}{q}$$

Using the dispersion relation

$$\begin{aligned} \omega &= cq \qquad \text{right-traveling wave } v_\varphi = c \\ \omega &= -cq \qquad \text{left-traveling wave } v_\varphi = -c \end{aligned}$$

More generally, the linear wave equation allows solutions

$$u(x,t) = u(x \pm ct) \qquad \text{for arbitrary functions } u$$

The function $u$ is determined by the initial condition.
Simpler wave equation: waves travel only in one direction

$$\frac{\partial u}{\partial t} = c\frac{\partial}{\partial x}u$$

For a numerical treatment we need to handle also the spatial derivatives

**Finite-difference method:**
Introduce a spatial grid $x_j = j\Delta x$ with $u_j = u(x_j)$, $0 \le j \le N$, and approximate the spatial derivatives by Taylor expansion

$$u_{j\pm1} = u_j \pm \Delta x\frac{\partial}{\partial x}u_j + \frac{1}{2}\Delta x^2\frac{\partial^2}{\partial x^2}u_j + \mathcal{O}(\Delta x^3)$$

- First derivative

$$u_{j+1} - u_{j-1} = 2\Delta x\frac{\partial}{\partial x}u_j + \mathcal{O}(\Delta x^3) \qquad \frac{\partial u}{\partial x} = \frac{1}{2\Delta x}(u_{j+1} - u_{j-1}) + \mathcal{O}(\Delta x^2)$$

  This is a central difference scheme (cf. temporal derivative in the CN).

- Second derivative

$$\begin{aligned} u_{j+1} + u_{j-1} &= 2u_j + \Delta x^2\frac{\partial^2}{\partial x^2}u_j + \mathcal{O}(\Delta x^3) \\ \frac{\partial^2 u_j}{\partial x^2} &= \frac{1}{\Delta x^2}(u_{j+1} - 2u_j + 2_{j-1}) + \mathcal{O}(\Delta x^2) \end{aligned}$$

## 10.1 Unidirectional wave equation

Consider simplified, uni-directional wave equation

$$\frac{\partial u}{\partial t} = c\frac{\partial}{\partial x}u$$

in finite differences

$$\frac{\partial u_j}{\partial t} = c\frac{1}{2\Delta x}(u_{j+1} - u_{j-1})$$

The coupled system can be written in terms of the vector

$$\mathbf{u}(t) = \begin{pmatrix} u_0 \\ u_1 \\ ... \\ u_N \end{pmatrix}$$

as

$$\frac{\partial}{\partial t}\mathbf{u} = c\mathbf{D}\mathbf{u} \equiv \mathbf{F}(\mathbf{u})$$

with

$$\mathbf{D} = \begin{pmatrix} 0 & \frac{c}{2\Delta x} & & & \\ -\frac{c}{2\Delta x} & 0 & \frac{c}{2\Delta x} & & \\ & -\frac{c}{2\Delta x} & 0 & ... & \\ & & ... & 0 & \frac{c}{2\Delta x} \\ & & & -\frac{c}{2\Delta x} & 0 \end{pmatrix}$$

**Note:**

- In finite domains we need to consider also boundary conditions: see sec.10.3

Now the coupled equations can be treated with one of the time-stepping routines discussed previously.

**Forward Euler:**

$$u_j^{n+1} = u_j^n + \Delta t\frac{c}{2\Delta x}\left(u_{j+1}^n - u_{j-1}^n\right) + \Delta t\,\mathcal{O}(\Delta t, \Delta x^2)$$

**Note:**

- superscript indicates time step, subscript spatial grid point
- forward Euler is first-order in time, second-order in space: (1,2)-scheme

**Neumann Stability Analysis:**

Ansatz for the solution of the finite-difference scheme is similar to our previous stability analysis,

$$u_j^n = \mathcal{U}_0\, e^{iqx_j}\, z^n$$

$$z = 1 + \frac{1}{2}\frac{c\Delta t}{\Delta x}\left(e^{iq\Delta x} - e^{-iq\Delta x}\right) = 1 + i\frac{c\Delta t}{\Delta x}\,\sin(q\Delta x)$$

The iteration grows if $|z|^2 > 1$

$$|z|^2 = 1 + \left(\frac{c\Delta t}{\Delta x}\right)^2 \sin^2 q\Delta x > 1 \qquad \text{for all } \Delta t \text{ as long as } q\Delta x \neq 0, \pi$$

**Thus:**

- Since all wavenumbers
$$-\frac{\pi}{\Delta x} \leq q \leq \frac{\pi}{\Delta x}$$
are possible on a grid with grid spacing $\Delta x$, forward Euler is *unstable* for the wave equation for *any* $\Delta t$.

- The fastest growing modes have large wavenumber:
$$q = \frac{\pi}{2\Delta x} \qquad \Leftrightarrow \qquad \ell = \frac{2\pi}{q} = 4\Delta x.$$
  - Even if the truncation error and the round-off error have only very small contributions for modes with those high wavenumbers these modes will eventually take over.
  - As the spatial grid is refined the temporal growth rate increases; it diverges for $\Delta x \to 0$.
  - For a fixed grid size the divergence can be delayed by taking a sufficiently small time step: the longer the desired duration $T$ of the run the smaller $\Delta t$ has to be chosen.

- Forward Euler is not suitable to solve wave equations for extended periods of time.

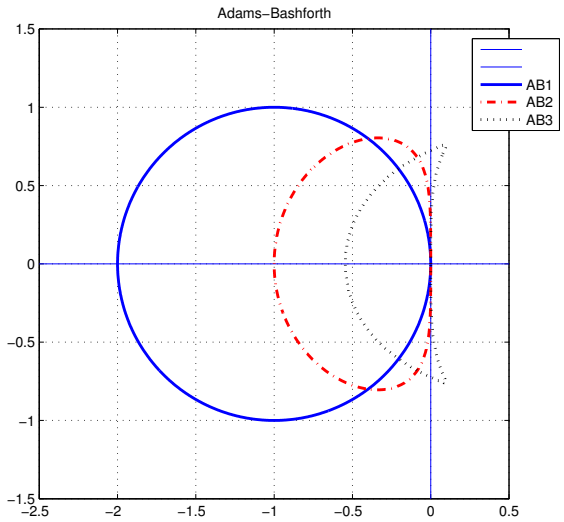
**Demo:**

- `wave_demo.m` of Euler dependence on dx:
  - xmax=10 nx=200 dt=0.01 tmax=25 divergence by t=18. fastest growing mode has wavelength 4dx.
  - then nx=300 , divergence by t=8.
  - plot also estimate for z to compare with Euler

**Note:**

- Inserting the Fourier ansatz $u = \mathcal{U}(t)e^{iqx}$ in the wave equation we obtain
$$\frac{d\mathcal{U}}{dt} = icq\mathcal{U} \qquad \Leftrightarrow \qquad \frac{du}{dt} = \lambda u \qquad \text{with } \lambda = icq$$
Thus, we can use our previous stability results: we know that forward Euler is not stable for $\lambda \in i\mathbb{R}$ and the growth rate grows with increasing wavenumber

$q$.




Suitable schemes:

- explicit schemes: AB3, RK4
- implicit schemes: BE, BD2, CN

**Dimensional analysis:**

In general, any stability limits will be of the form

$$\frac{c\Delta t}{\Delta x} \leq \Lambda_{max}$$

with $\Lambda_{max}$ a dimensionless number characterizing the scheme (CFL-condition after Courant, Friedrich, and Lewy).

For small $\Delta x$ the system is *stiff*:

- Stability limit: $\Delta t_{max} \to 0$ for $\Delta x \to 0$
  Fine grids allow high wavenumbers: the code needs to be stable for those wavenumbers even if the solution does not contain energy in those modes. Thus, as the spatial grid is refined the time step also has to be refined even if the temporal accuracy does not require it, since the dynamics of the physical problem do not get faster with the refinement of the grid.
- The slowest modes are associated with the smallest wavenumbers $q = \frac{2\pi}{L}$. The range of time scales is therefore related to the range defined by $\frac{1}{L}$ and $\frac{1}{\Delta x}$, which is large for $\Delta x \ll L$.
- Algebraically: the largest eigenvalues of the differentiation matrix D are $\mathcal{O}(\frac{1}{\Delta x})$; they become large for large $N$.

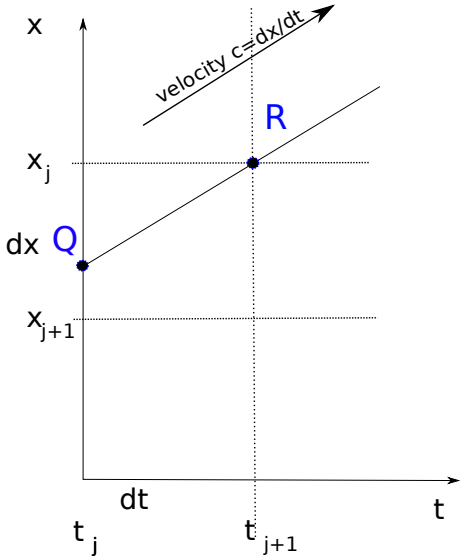



Figure 17: Space-time diagram for the information flow by the wave.

**Domain of Dependence**

The wave propagates with speed $c$. The information for the new value at point $R$ comes from point $Q$:

- $\Delta x > c\Delta t$: the point $Q$ is between $x_j$ and $x_{j+1}$ and the information about $R$ (needed for updating the solution at $x_j$) is available from the grid points involved in the schemes, $x_j$ and $x_{j+1}$.
- $\Delta x < c\Delta t$: the point $Q$ is beyond $x_{j+1}$ and the information about $R$ is *not available* from $x_j$ and $x_{j+1}$: the scheme 'does not know' how to update $R$: *unstable*.

**Note:**

- The domain of dependence explains qualitatively why explicit schemes have a maximal time step limit $\Delta t_{max}$.
- For implicit schemes the matrix inversion couples $x_j$ effectively to *all* grid points (cf. (23) below)

## 10.2 Diffusion Equation

To assess the stability of numerical schemes for the diffusion equation

$$\frac{\partial u}{\partial t} = D\frac{\partial^2 u}{\partial x^2}$$

make again a Fourier ansatz

$$u = \mathcal{U}(t) e^{iqx}$$

and obtain ODEs for each Fourier mode,

$$\frac{d\mathcal{U}}{dt} = \lambda \mathcal{U} \qquad \text{with } \lambda = -Dq^2 \in \mathbb{R}$$

Since $\lambda \in \mathbb{R}$ we expect most explicit schemes (e.g. forward Euler) to be stable up to some $\Delta t_{max}$.

Dimensional analysis yields in this case

$$\Delta t_{max} = \frac{1}{D} \Delta x^2 \Lambda_{max} = \mathcal{O}(\Delta x^2)$$

**Note:**

- The scaling of $\Delta t$ with $\Delta x$ in the stability limit is quite restrictive. Numerically, the diffusion equation is even more stiff than the wave equation: using implicit schemes is almost required.
- For higher-order problems the scaling becomes yet more restrictive

$$\frac{\partial u}{\partial t} = \frac{\partial^n u}{\partial x^n} \qquad \Rightarrow \qquad \Delta t_{max} \leq \mathcal{O}(\Delta x^n).$$

Illustrate this stability limit explicitly:

$$\frac{\partial}{\partial t} \mathbf{u} = \mathbf{D}^{(2)} \mathbf{u}$$

with

$$\mathbf{D}^{(2)} = \begin{pmatrix} -\frac{2}{\Delta x^2} & \frac{1}{\Delta x^2} & & & \\ \frac{1}{\Delta x^2} & -\frac{2}{\Delta x^2} & \frac{1}{\Delta x^2} & & \\ & \frac{1}{\Delta x^2} & -\frac{2}{\Delta x^2} & \frac{1}{\Delta x^2} & \\ & & \cdots & \cdots & \cdots \\ & & & \frac{1}{\Delta x^2} & -\frac{2}{\Delta x^2} \end{pmatrix}$$

**Forward Euler:**

$$u_j^{n+1} = u_j^n + D \frac{\Delta t}{\Delta x^2} \left( u_{j+1}^n - 2u_j^n + u_{j-1}^n \right) + \Delta t \, \mathcal{O}(\Delta t, \Delta x^2)$$

Neumann analysis

$$u_j^n = \mathcal{U}_0 e^{iqx_j} \, z^n$$

$$z = 1 + D \frac{\Delta t}{\Delta x^2} \left( e^{iq\Delta x} - 2 + e^{-iq\Delta x} \right) = 1 + 2D \frac{\Delta t}{\Delta x^2} (\cos(q\Delta x) - 1) < 1$$

Since $z \leq 1$ for all $\Delta t$, stability requires

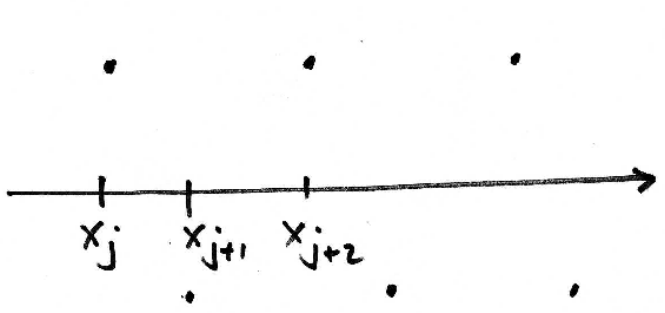
$$z \geq -1$$

$z$ most negative for $q\Delta x = \pi$. Therefore we get the requirement

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

**Note:**

- The mode with largest growth rate has $q\Delta x = \pi$




The excitation of the highest wavenumber is typical for numerical instability $z < -1$ also temporal oscillations like in the instability of ODEs

**Crank-Nicholson:**

$$u_j^{n+1} = u_j^n + \frac{1}{2}D\frac{\Delta t}{\Delta x^2}\left(u_{j+1}^n - 2u_j^n + u_{j-1}^n\right) + \frac{1}{2}D\frac{\Delta t}{\Delta x^2}\left(u_{j+1}^{n+1} - 2u_j^{n+1} + u_{j-1}^{n+1}\right)$$

$$-D\frac{\Delta t}{2\Delta x^2}u_{j+1}^{n+1} + (1 + D\frac{\Delta t}{\Delta x^2})u_j^{n+1} - D\frac{\Delta t}{2\Delta x^2}u_{j-1}^{n+1} = u_j^n + D\frac{\Delta t}{2\Delta x^2}\left(u_{j+1}^n - 2u_j^n + u_{j-1}^n\right) \quad (23)$$

in terms of u

$$\begin{pmatrix} \frac{1}{\Delta t} + \frac{D}{\Delta x^2} & \frac{-D}{2\Delta x^2} & & \\ \frac{-D}{2\Delta x^2} & \frac{1}{\Delta t} + \frac{D}{\Delta x^2} & \frac{-D}{2\Delta x^2} & \\ & \cdots & \cdots & \cdots \\ & & \frac{-D}{2\Delta x^2} & \frac{1}{\Delta t} + \frac{D}{\Delta x^2} \end{pmatrix} \mathbf{u}^{n+1} = \begin{pmatrix} \frac{1}{\Delta t} - \frac{D}{\Delta x^2} & \frac{D}{2\Delta x^2} & & \\ \frac{D}{2\Delta x^2} & \frac{1}{\Delta t} - \frac{D}{\Delta x^2} & \frac{D}{2\Delta x^2} & \\ & \cdots & \cdots & \cdots \\ & & \frac{D}{2\Delta x^2} & \frac{1}{\Delta t} - \frac{D}{\Delta x^2} \end{pmatrix} \mathbf{u}^n$$

**Notes:**

- requires solving large tridiagonal matrix
- for nonlinear problems issue with inverting matrix in each time step and in each step of Newton iteration, but likely the matrix is even larger than for the ODEs.

In general, compared to ODEs issue for PDEs:

- problems typically stiff due to large number of grid points

## 10.3 Boundary Conditions

### Diffusion equation

Consider example

$$\frac{\partial}{\partial t}u = \frac{\partial^2}{\partial x^2}u \qquad 0 \leq x \leq L$$

with two types of boundary conditions

$$\begin{aligned} u(x=0,t) &= g_0(t) \\ \alpha u(x=L,t) + \beta\frac{\partial}{\partial x}u(x=L,t) &= g_L(t) \end{aligned}$$

At $x = 0$ Dirichlet boundary condition (e.g. fixed temperature).
At $x = L$ Robin boundary condition. For $\alpha = 0$ and $g_L = 0$ no-flux (Neumann) boundary condition $\frac{\partial}{\partial x}u = 0$.

For simplicity assume forward Euler scheme

$$u_j^{n+1} = u_j^n + \frac{\Delta t}{\Delta x^2}\left(u_{j+1}^n - 2u_j^n + u_{j-1}^n\right)$$

with $x_0 = 0$ and $x_N = L$.

1. $u(x=0,t)$ is given
   - apply PDE only to grid points $j = 1, ...$
   - $u_0 = g(t)$
2. $u(x=L,t)$ is not given explicitly
   - discretize boundary condition

     $$\alpha u_N + \beta\frac{1}{2\Delta x}(u_{N+1} - u_{N-1}) = g_L(t)$$

   - $u_{N+1}$ is a *fictitious* point outside the computational domain
   - use PDE to update $u_j$ for $j = ..., N-1$.
   - to update $u_N$ we need $u_{N+1}$. We get $u_{N+1}$ from the boundary condition.

### Wave Equation

$$\frac{\partial}{\partial t}u = \frac{\partial}{\partial x}u \qquad 0 \leq x \leq L$$

For wave equations not all boundary conditions lead to well-posed problems:

- Wave propagates to the left $\Rightarrow$ $u_j$ obtains its information from $u_{j+1}$ **but not** from $u_{j-1}$
- $u_N$ gets its information from $u_{N+1}$ outside the computational domain: need boundary condition at $x = L$, e.g.

$$u_N = g(t)$$

- $u_0$ gets its information from $u_1$: $u_0$ completely determined from interior of the domain: **no boundary condition** allowed. Use *one-sided* difference for updating $u_0$

$$u_0^{n+1} = u_0^n + \frac{\Delta t}{\Delta x}\,(u_1^n - u_0^n)$$

**Notes:**

- if a numerical boundary condition is posed where PDE does not allow a boundary condition scheme becomes *unstable* (e.g. oscillations propagate into the interior from that boundary)
- in systems of wave equations waves may propagate in both directions
  pose boundary condition on the 'component' of the wave that enters the computational domain (*incoming characteristic variable*) but not on the 'component' that leaves the domain (*outgoing characteristic variable*)

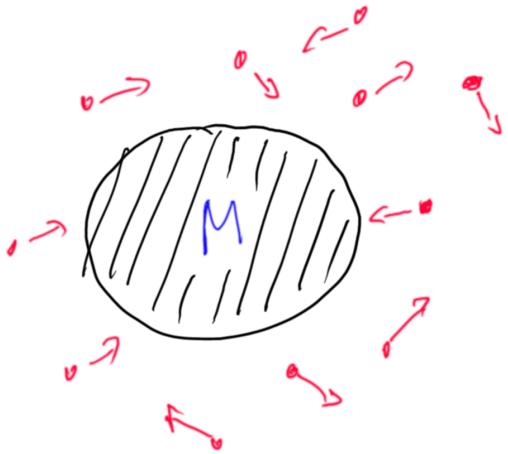
# 11 Stochastic Differential Equations

Stochastic differential equations are important in a wide range of applications in science, engineering, finance, ... To mention just a few examples:

1. Brownian motion of a dust speck
   - The dust speck is bombarded by with many, many molecules in rapid succession. It is impossible to model the motion of all these molecules. What to do?
   - The motion of the molecules is so disordered that collisions are essentially independent of each other over any macroscopic time scale

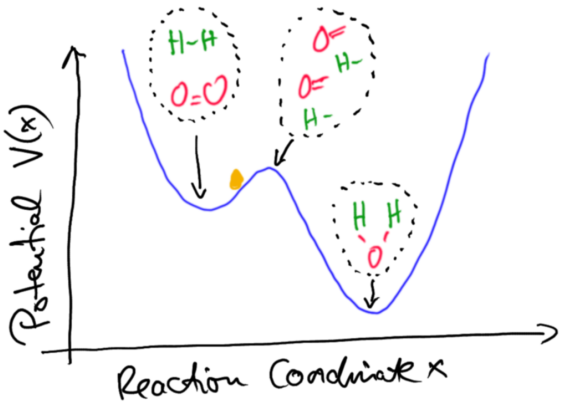
$$\gamma \frac{dx}{dt} = F(x) + \xi(t)$$

- Model the collisions by $\xi(t)$: fluctuates extremely rapidly and unpredictably, i.e. $\xi(t_1)$ and $\xi(t_2)$ are uncorrelated if $t_1 \neq t_2$.




2. Chemical reactions

   - Molecules are pushed by other molecules: model as 'noise' $\xi(t)$.
   - Molecules interact with each other
     - The molecules may have to be rearranged or chemical bonds may have to be broken before new bonds can be formed: barriers may have to be overcome.
     - for close distances after rearrangement: attraction through binding
   - Model the reacting molecules as 'particle' in a potential:
     The position in the potential is not really a position of the molecule but rather a reaction coordinate that describes the rearranging of the molecules.




     Many reactions are activated, i.e. for the reaction to proceed an activation energy barrier has to be surpassed, which requires noise.

3. Ion channels

   - Each channel, whcih essentially consists of a few molecules in a membrane that can change their shape depending on the electric field, has a certain probability to open or close during a given time interval

     $$Closed \quad \xrightarrow[\text{probability } \alpha]{} \quad Open$$

     $$Closed \quad \xleftarrow[\text{probability } \beta]{} \quad Open$$

   - Averaging over many channels one gets for the fraction $n(t)$ of open channels

     $$\frac{dn}{dt} = \alpha\,(1-n) - \beta\,n$$

   - But at any given time the actual number $N$ of open channels probably differs from the mean; relative to the mean deviations are larger for smaller number of channels
     $n(t)$ will fluctuate around $\bar{n}$. We can model these fluctuations with a 'noise' term

     $$\frac{dn}{dt} = \alpha\,(1-n) - \beta\,n + \zeta$$

     **Note:** we would have to make sure that noise does not make push $n$ out of the range $[0, 1]$.

4. Spreading of infectious diseases

   - SIS-model:
     everybody has a certain probability to contract disease
     and certain probability to recover

     $$\text{susceptible} \quad \xrightarrow[\text{probability } \alpha]{} \quad \text{infected}$$

     $$\text{susceptible} \quad \xleftarrow[\text{probability } \beta]{} \quad \text{infected}$$

   - SIRS-model: resistent period after recovery
   - The mean of the healthy/susceptible fraction of the population satisfies again a differential equation.
   - The fluctuations can be modeled with 'noise'.

5. Stock Market

   Many independently buying and selling agents lead to fluctuations of the price.

We need to discuss a few mathematical concepts:

**Random Variables** $\xi$

Consider repeating the same experiment many times:
each time the outcome $\xi$ has a certain probability $P(\xi)$ to occur
the mean of $\xi$ after many repeats will be

$$E(\xi) \equiv < \xi > = \int \xi P(\xi)\, d\xi$$

Example: Gaussian distribution

$$P(\xi) = N(\mu, \sigma^2) \equiv \frac{1}{\sqrt{2\pi\sigma^2}} e^{\frac{-(x-\mu)^2}{2\sigma^2}} \tag{24}$$

with

$$\text{mean} \quad E(\xi) = \mu \qquad\qquad \text{variance} \quad E\left((\xi-\mu)^2\right) = \sigma^2 \tag{25}$$

**Notes:**

- The Gaussian distribution is very common for the collection of many independent events due to the *Central Limit Theorem*:
  In the limit of large $N$ the probability distribution function for the sum $S_N = \sum_{i=1}^{N} \xi_i$ of $N$ *identically distributed independent variables* is given by a Gaussian distribution. For instance, the collective effect of a large number of kicks in Brownian motion is Gaussian distributed.
  The Gaussian has

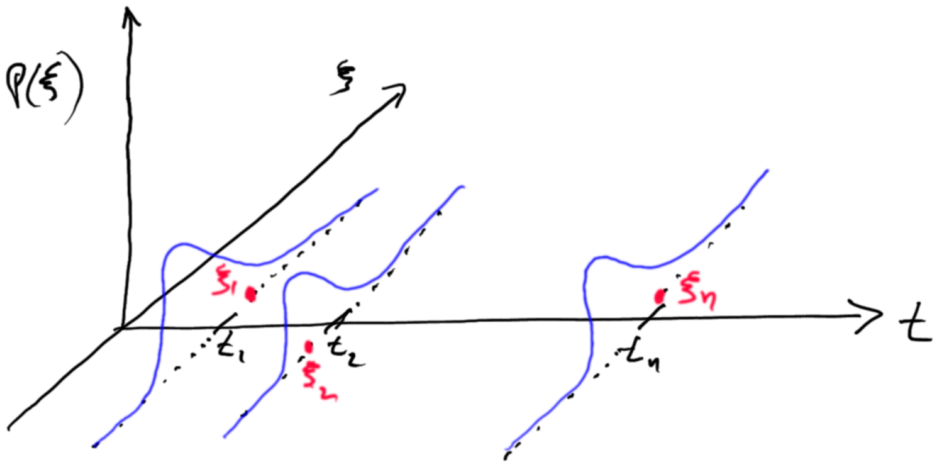
  $$\mu = N \underbrace{E(\xi_i)}_{\text{mean of } \xi_i} \qquad \sigma^2 = N \underbrace{E((\xi_i-\mu)^2)}_{\text{variance of individual } \xi_i} \tag{26}$$

  Thus: the mean of the sum grows linearly with $N$ while the standard deviation $\sigma$ grows like $\sqrt{N}$.

  - the Gaussian distribution is also called *normal* distribution (`randn` in matlab).
  - the Gaussian is *unbounded*: arbitrarily large events are possible (although unlikely)
    The velocity distribution of a finite number of particles would always have to be bounded by the total energy: the Gaussian arises in the limit of large $N$ in which case the total energy diverges.

**Stochastic Process** $\xi(t)$:

For any value of the time $t$ the output of a stochastic process is given by a random variable $\xi(t)$. Thus, for any sequence $t_1, t_2, ...t_N$ one gets a sequence $\xi(t_1) \equiv \xi_1, \xi(t_2) \equiv \xi_2, ..., \xi(t_N) \equiv \xi_N$ of random variables.




Examples of stochastic processes:

- In a Gaussian process each element $\xi_i$ of the process is normally distributed,

$$P(\xi_i) = \frac{1}{\sqrt{2\pi\sigma^2}} e^{\frac{-(\xi_i-\mu)^2}{2\sigma^2}} \qquad \langle \xi_i \rangle = \mu \qquad \left\langle (\xi_i - \mu)^2 \right\rangle = \sigma^2$$

  More generally, the mean and the variance could depend on time; in that case $\mu$ and $\sigma$ would be replaced by $\mu_i$ and $\sigma_1$.
  The random variables $\xi_i \equiv \xi(t_i)$ for different times *may or may not be independent* of each other.

- Markov processes
  In a discrete Markov process the probability of any event depends only on the previous event

$$P(\xi_n, \xi_{n-1}, \xi_{n-2}, ...) = P(\xi_n, \xi_{n-1}) = P(\xi_n|\xi_{n-1})\, P(\xi_{n-1})$$

  with $P(\xi_n|\xi_{n-1})$ being the conditional probability to get $\xi_n$ in the next step if the value at the previous step was $\xi_{n-1}$.
  A Markov process is somewhat like a first-order ODE in the sense that only a single initial condition is needed.

- Wiener process $W(t)$
  The Wiener process models Brownian motion: in Brownian motion the collisions are assumed to be independent of each other: each collision changes the velocity of the particle randomly

  - Successive velocities are independent,
  - The difference in the particle's position is proportional to the velocity $\Rightarrow$ the increments of the position are independent.

– But successive positions are not independent of each other.

More specifically:
If $W(t_n)$ and $W(t_{n+1})$ are random variables their difference is also a random variable.
The Wiener process $W(t)$ is a stochastic process that satisfies

– $W(0) = 0$
– the increments $dW(t_n) = W(t_{n+1}) - W(t_n)$ during a time interval $t_{n+1} - t_n$ are **Gaussian** distributed according to

$$P\left(W(t_{n+1}) - W(t_n)\right) = N(0, t_{n+1} - t_n).$$

Thus, the variance of an increment grows linearly with its time difference.

– increments $dW(t_n)$ and $dW(t_m)$ are *independent* random variables if their time intervals do not overlap
$\Rightarrow W(t)$ is a Markov process since $W(t_{n+1}) = W(t_n) + dW(t_n)$.

The 2-point joint probabilities are therefore given by

$$P(w_2, t_2; w_1, t_1; 0, 0) = P(w_2, t_2 | w_1 t_1) P(w_1, t_2 | 0, 0) \qquad \text{for } t_2 > t_1$$

with

$$P(w_2, t_2 | w_1, t_1) = \frac{1}{\sqrt{2\pi\,(t_2 - t_1)}} e^{-\frac{1}{2}\frac{(w_2 - w_1)^2}{(t_2 - t_1)}}$$

Later we will need the autocorrelation function

$$\begin{aligned}
\langle W(t_2) W(t_1) | 0, 0 \rangle &= \int dw_1 dw_2\, w_2 w_1\, P(w_2, , t_2; w_1 t_1; 0, 0) \qquad \text{for } t_2 > t_1 \\
&= \int dw_1 dw_2 \left( \underbrace{w_2 - w_1}_{\text{odd}} + w_1 \right) w_1\, \underbrace{P(w_2, t_2 | w_1 t_1)}_{\text{even}}\, P(w_1, t_2 | 0, 0) \\
&= \int dw_1\, w_1^2\, P(w_1, t_2 | 0, 0) = t_1
\end{aligned}$$

In words: the expectation value averages over all possible points $w_1$ at $t_1$ and $w_2$ at $t_2$. Consider first a fixed point $w_1$ at $t_1$. The random walks for any $t > t_1$ are evenly distributed about $w_1$; on average all these walks yield therefore $w_1$. This leaves then the average of $w_1^2$ which is $t_1$.

For general $t_1$ and $t_2$ this can be written as

$$\langle W(t_2) W(t_1) | 0, 0 \rangle = \min(t_1, t_2)$$

Somewhat more condensed we could have written

$$\langle W(t_2) W(t_1) | 0, 0 \rangle = \left\langle \underbrace{\left(W(t_2) - W(t_1)\right)}_{dW(t_1)} W(t_1) \right\rangle + \left\langle \left(W(t_1)\right)^2 \right\rangle$$

Because the increment $dW(t_1)$ is independent of $W(t_1)$ the first term drops out since $\langle dW(t_1) \rangle = 0$, and the second term is the linear growth of the variance of the Wiener process.

## 11.1 Snippets of Ito Calculus

How do we deal with the stochastic differential equation (Langevin equation)

$$\frac{dy}{dt} = F(y) + g(y)\,\xi(t) \tag{27}$$

What is the issue?

We want such a Langevin equation to model, e.g., Brownian motion. Thus, we imagine a noise term $\xi$ for which the solution $y(t)$ to the simpler equation

$$\frac{dy}{dt} = \xi(t)$$

gives Brownian motion, i.e. a Wiener process.

The noise would therefore have to be the derivative of the Wiener process and the Wiener process would have to satisfy this differential equation. However, the Wiener process is nowhere differentiable: if the derivative existed we could compute the expectation value of its square. Consider

$$\lim_{\Delta t \to 0} \left\langle \left( \frac{W(t+\Delta t) - W(t)}{\Delta t} \right)^2 \right\rangle = \lim_{\Delta t \to 0} \frac{1}{\Delta t^2} \Delta t = \lim_{\Delta t \to 0} \frac{1}{\Delta t} \qquad \text{NO!}$$

Thus, the limit $\Delta t \to 0$ does not exist and the derivative $dy/dt = dW/dt$ does not exist.

**Demo**: Show solutions with decreasing $\Delta t$ demonstrating that the Wiener process is not differentiable.

Similarly, we do not expect the solutions of the more general equation (27) to be differentiable, i.e. we cannot define the evolution in terms of the usual derivative.

To avoid having the derivates write the Langevin equation in differential form

$$dy = F(y(t))dt + g(y(t))\,dW(t) \tag{28}$$

where we take $dW(t)$ to be the increment of the Wiener process $W(t)$ over an infinitesimal time interval: $dW(t) = W(t+dt) - W(t)$.

Solving (28) requires to integrate this equation, which wold require less smoothness than taking a derivative.

How do we integrate over $dW(t)$? Try to define an integral $\int g(y(t))dW(t)$ over the differential $dW$ by discretizing the time and considering for arbitrary continuous functions $g$ the sum

$$\lim_{N\to\infty} \underbrace{\sum_{i=1}^{N} g\left(y(t_i^*)\right)\, dW(t_i)}_{S_N} \qquad \text{with} \quad t^* \in [t_i, t_{i+1}].$$

For different realizations of the Wiener process the sum has a different value. To make sense of the limit we therefore average over multiple realizations and evaluate the sums in the *mean-square limit*:
For a sequence $S_N$ of random variables we define the *mean-square limit* via

$$ms - \lim_{N\to\infty} S_N = S \qquad \Leftrightarrow \qquad \lim_{N\to\infty} \left\langle (S_N - S)^2 \right\rangle \equiv \lim_{N\to\infty} \int P(S_N)\,(S_N - S)^2 dS_N = 0.$$

We can now define an integral involving the differential $dW(t)$ by

$$\int g(t)\, dW(t) = ms - \lim_{N\to\infty} \sum_{i=1}^{N} g(t_i^*)\ (W(t_i) - W(t_{i-1})) \qquad \text{with} \quad t^* \in [t_{i-1}, t_i]$$

If $W(t)$ was a *continuosly differentiable* function of $t$ one could write $dW(t) = \frac{dW}{dt}dt$ and one would obtain the usual definition of an integral in terms of a Riemann sum.

If $W(t)$ had *bounded variation* this would be a *Riemann-Stieltjes* integral and one could show that the value of the sum is independent of the specific choice of $t^* \in [t_i, t_{i+1}]$. In that case also the usual integration rules result.

In the Wiener process, however, the increments $dW(t_i) = W(t_{i+1}) - W(t_i)$ are Gaussian distributed and are therefore *unbounded*. In general the sum therefore depends on the specific choice of the value $t^*$ at which $g(t)$ is evaluated.
Consider the specific example $g(t) = W(t)$ and write the expectation value of the partial sum

$$S_N = \left\langle \sum_{i=1}^{N} W(t_i^*)\ (W(t_i) - W(t_{i-1})) \right\rangle = \sum_{i=1}^{N} \min\left(t_i^*, t_i\right) - \min\left(t^*, t_{i-1}\right)$$

Consider

$$t_i^* = \alpha t_i + (1-\alpha)\, t_{i-1} \qquad 0 < \alpha < 1$$

Then

$$S_N = \sum_{i=1}^{N} \alpha t_i + (1-\alpha)\, t_{i-1} - t_{i-1} = \alpha\,(t_N - t_0)$$

Thus, $S_N$ and with it the integral depends on $\alpha$, i.e. the choice of $t^*$. Commonly one considers the two possibilities

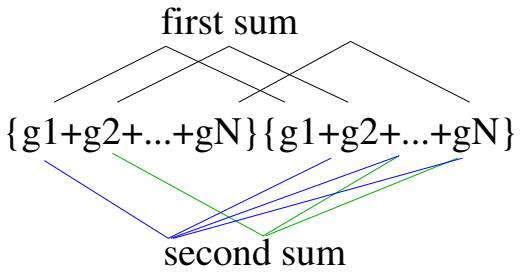
1. Ito :
$$\int g(y(t))\, dW(t) \equiv \lim_{N\to\infty} \sum_{i=1}^{N} g(y(t_{i-1}))\, (W(t_i) - W(t_{i-1}))$$
The function $g(y(t))$ is evaluated at *left end-point* of each interval: the 'kick' of the Wiener process (cf. Brownian motion) has a strength corresponding to the *previous* position of the particle.
The Ito integral is defined for *non-anticipating* functions $g(y(t))$, i.e. $g(y(t_{i-1}))$ must be statistically independent of the increment $W(t_i) - W(t_{i-1})$.

2. Stratonovich:
$$\int g(y(t))\, dW(t) \equiv \lim_{N\to\infty} \sum_{i=1}^{N} \frac{1}{2}\, (g(y(t_{i-1})) + g(y(t_i)))\, (W(t_i) - W(t_{i-1}))$$
The function $g(y(t))$ is evaluated at the *mid-point* of the interval: the kick of the Wiener process has an effective strength corresponding to a position at the 'middle of the kick'.

The two integrals give *different* results. For stochastic integrals one therefore always has to define which of the two interpretations is meant. In the following we will assume the Ito interpretation, i.e.
$$\lim_{N\to\infty} \langle \left[ \int g(y(t)) dW(t) - \sum_{i=1}^{N} g(y(t_{i-1}))\ (W(t_i) - W(t_{i-1})) \right]^2 \rangle = 0$$

**Basic Rules**

1. $dW^2 = dt$, $dW^{2+n} = 0$ for $n > 0$
This means that for any non-anticipating continuous function one has
$$\int_0^t g(y(t'))\ [dW(t')]^2 \equiv ms{-}\lim_{N\to\infty} \sum_{i}^{N} g(y(t_{i-1}))\ [W(t_i) - W(t_{i-1})]^2 = \int_0^t g(y(t'))\, dt'$$
The proof makes use of the independence of successive increments $dW(t_{i+1})$ and $dW(t_i)$ and of increments in $W$ and earlier values of $g(t)$.
To prove this identity consider the appropriate limit
$$\lim_{N\to\infty} \left\langle \left\{ \sum_{i=1}^{N} G_{i-1} \left[ \underbrace{\Delta W_i^2}_{(W_i - W_{i-1})^2} - \Delta t_i \right] \right\}^2 \right\rangle = \lim_{N\to\infty} \left\langle \sum_{i=1}^{N} G_{i-1}^2 \left(\Delta W_i^2 - \Delta t_i\right)^2 + 2 \sum_{i=1}^{N} \sum_{j=i+1}^{N} G_{i-1} G_{j-1} \left(\Delta W_i^2 - \Delta t_i\right) \left(\Delta W_j^2 - \Delta t_j\right) \right\rangle$$




The sum consists of all possible combinations of the terms in each of the two sums; thus, the indices of each sum can be thought of as the indices of a 2-dimensional symmetric matrix

$$\begin{pmatrix} g_1g_1 & g_1g_2 & g_1g_3 & & g_1g_N \\ g_2g_1 & g_2g_2 & & & \\ & & & & \\ & & & & \\ g_Ng_1 & & & & g_Ng_N \end{pmatrix}$$

where $g_i = G_{i-1}\left[\Delta W_i - \Delta t_i\right]$. The overall sum is the then sum over all matrix elements: the first term represents the diagonal elements and the second term the off-ciagonal terms.

Term with single sum:

$G$ is non-anticipating $\Rightarrow G_{i-1}$ and $\Delta W_i$ are statistically independent

$$\left\langle G_{i-1}^2\left(\Delta W_i^2 - \Delta t_i\right)^2\right\rangle = \left\langle G_{i-1}^2\right\rangle \left\langle \left(\Delta W_i^2 - \Delta t_i\right)^2\right\rangle$$

Since $W$ is Gaussian distributed the fourth moment is related to the second moment

$$\begin{array}{lll} \left\langle \Delta W_i^4\right\rangle & = & 3\left\langle \Delta W_i^2\right\rangle^2 = 3\left\langle (W_i - W_{i-1})^2\right\rangle^2 = \\ & \underbrace{=}_{\text{Wiener process}} & 3\left\{t_i - 2t_{i-1} + t_{i-1}\right\}^2 = 3\Delta t_i^2 \end{array}$$

and

$$\left\langle G_{i-1}^2\left(\Delta W_i^2 - \Delta t_i\right)^2\right\rangle = \left\langle G_{i-1}^2\right\rangle \left\{3\Delta t_i^2 - 2\Delta t_i^2 + \Delta t_i^2\right\} = 2\left\langle G_{i-1}^2\right\rangle \Delta t_i^2$$

Therefore

$$\lim_{N\to\infty}\left\langle \sum_{i=1}^{N} G_{i-1}^2\left(\Delta W_i^2 - \Delta t_i\right)^2\right\rangle = 2\lim_{N\to\infty}\left\{\underbrace{\sum_{i=1}^{N}\left\langle G_{i-1}^2\right\rangle \Delta t}_{=\mathcal{O}(1)}\cdot\Delta t\right\} = 0$$

Term with double sum:

$j > i \Rightarrow G_{i-1}G_{j-1}\left(\Delta W_i^2 - \Delta t_i\right)$ is statistically independent of $\Delta W_j^2 - \Delta t_j$

$$\left\langle G_{i-1}G_{j-1}\left(\Delta W_i^2 - \Delta t_i\right)\left(\Delta W_j^2 - \Delta t_j\right)\right\rangle = \left\langle G_{i-1}G_{j-1}\left(\Delta W_i^2 - \Delta t_i\right)\right\rangle \underbrace{\left\langle \Delta W_j^2 - \Delta t_j\right\rangle}_{=0}$$

Thus

$$\lim_{N\to\infty}\left\langle \left\{ \sum_{i=1}^{N} G_{i-1}\left[ \underbrace{\Delta W_i^2}_{(W_i - W_{i-1})^2} - \Delta t_i \right] \right\}^2 \right\rangle = 0$$

The proof of $dW^{2+n} = 0$ is analagous. It makes use of the higher cumulants and is more cumbersome.

(a) **Note**:

- $dW^2 = dt$ corresponds to the property of Brownian motion: the variance of a diffusing particle grows linearly with time: $\langle x(t)^2\rangle \propto t$
  (cf. (26): $\sigma \propto \sqrt{N}$ where in each step a random increment step is *added* to the current position.)
- In this sense $dW$ is a differential of order $\frac{1}{2}$.

2. Integration of polynomials
   Consider only the simplest cases

   (a)
   $$\int_{t_1}^{t_2} dW(t) = W(t_2) - W(t_1)$$
   i.e. summing up the increments of the Wiener process (=Brownian motion) gives the distance traversed by the Brownian particle.

   (b) Now consider
   $$\begin{aligned} d(W(t)^2) &= (W(t+dt)^2 - W(t)^2) = \underbrace{(W(t)+dW)^2}_{dW \text{ is the increment of } W \text{ during } dt} - W(t)^2 = \\ &= 2W(t)dW + dW^2 = 2W(t)dW + dt \end{aligned}$$
   thus
   $$\int_0^t W(t')dW = \frac{1}{2}\left(W(t)^2 - \underbrace{W(0)^2}_{=0}\right) - \frac{1}{2}(t-0) = \frac{1}{2}W(t)^2 - \frac{1}{2}t$$

3. **Ito's formula**
   Consider a function $v(y)$ where $y$ satisfies the stochastic differential equation (28). What is the differential of $v\left(y(t,W)\right)$?

$$\begin{aligned} dv \quad \underbrace{=}_{\text{Taylor expansion}} \quad & \frac{dv}{dy}dy + \frac{1}{2}\frac{d^2v}{dy^2}\left(dy\right)^2 + h.o.t. = \\ = \quad & v'\left(F(y)dt + g(y)dW\right) + \frac{1}{2}v''\left[F(y)dt + g(y)dW\right]^2 + h.o.t. = \\ = \quad & \left(v'F + \frac{1}{2}v''g^2\right)dt + v'gdW \qquad \text{using } (dW)^2 = dt \end{aligned}$$

   **Note:**

   - Ito's formula is a stochastic equivalent of the chain rule for $v\left(y(t,W)\right)$.

**Note:**

- If $g$ depends on the dependent variable $y$ the noise is called multiplicative.
- If $g$ is independent of $y$ the noise is called additive.

**Example:**

Determine the exact solution for the linear Langevin equation with multiplicative noise

$$dy = ay(t)dt + by(t)\,dW \tag{29}$$

Without the noise term we would divide by $y$

$$\frac{dy}{y} = a\,dt$$

and by integrating both sides we would get

$$\ln y = at + C \qquad y = y_0 e^{at}.$$

Use this as a starting point and consider the differential of $\ln y$ and use Ito's formula

$$\begin{aligned} d\left(\ln y\right) &= \left(\frac{1}{y}\,ay + \frac{1}{2}(-\frac{1}{y^2}\,b^2y^2)\right)dt + \frac{1}{y}\,by\,dW \\ &= \left(a - \frac{1}{2}b^2\right)dt + b\,dW \end{aligned} \tag{30}$$

Thus, comparing with (29) we get

$$d\left(\ln y\right) = \frac{dy}{y} - \frac{1}{2}b^2\underbrace{dt}_{dW^2}$$

and the differential $d(\ln y)$ is not simply $dy/y$, but includes an additional term arising from $dW^2$. Consequently, $\int y^{-1}dy$ is not simply $\ln y$.

Because the coefficients of the differentials on the right-hand-side of (30) are constants we can integrate the expression for $d(\ln y)$ on both sides and get

$$\begin{aligned}\int_0^t d\left(\ln y\right) &= \ln y(t) - \ln y(0) = (a - \frac{1}{2}b^2)t + b\left(W(t) - \underbrace{W(0)}_{=0}\right) \\ y(t) &= y(0)e^{(a-\frac{1}{2}b^2)t + bW(t)}\end{aligned}$$

**Note:**

- The exact solution is still a stochastic process and depends, of course, on the realization $W(t)$
- Since $y$ is a stochastic variable one cannot simply use separation of variables and divide through by $y$ and integrate: the usual integration rules do not apply to the integral $\int y^{-1}dy$ if $y$ is a stochastic variable. For general stochastic variables $f$ the rules apply only to integrals like $\int df$.
- A strategy to solve such stochastic equations is to find a variable transformation that the takes the equation into one with constant coefficients. This is only possibly if $F(y)$ and $g(y)$ satisfy certain conditions.
- The multiplicative noise modifies the growth rate by $-\frac{1}{2}b^2$. This effect would be missed if we did not pay attention to Ito's formula.

## 11.2 Numerical Methods

For the numerical solution of stochastic differential equations two goals are possible

1. **Strong approximation**: pathwise approximation
   The numerical solution $y(t)$ approximates the exact solution $\tilde{y}$ for any given realization $W(t)$ of the Wiener process,

   $$\tau_s(t) \equiv E(|y(t) - \tilde{y}(t)|) = \frac{1}{N}\sum_{k=1}^{N}|y_k(t) - \tilde{y}_k(t)|.$$

   Here $y_k(t)$ is the numerical result one obtains for the $k^{th}-$realization of the Wiener process and analogously for the exact solution $\tilde{y}_k(t)$.
   Note: Add up the absolute value of the error of the different realizations to avoid error cancellation.

2. **Weak approximation**: approximation of expectation values
   The numerical solution does not approximate the exact solution for any given individual $W(t)$; it approximates only mean values of the exact solution. More precisely, for any $f(y)$ in a class of *test functions* the numerical solution approximates the expectation values of $f(y)$

$$\tau_m(t) =< f(\tilde{y}(t)) > - < f(y(t)) >$$

   Typically one would require convergence of the

   (a) mean: $f(y) = y$
   (b) variance: $f(y) = y^2$

**Notes:**

- For the strong approximation the numerical realizations $W(t)$ of the noise have to approximate the exact realizations.
- For the weak approximation the numerical noise can be quite different than the exact noise as long as it yields a $y(t)$ for which sufficiently many expectation values agree (e.g. mean, variance, higher moments $\langle (y(t))^m \rangle$).

### 11.2.1 Strong Approximation

#### i) Euler-Maruyama Scheme

Consider

$$dy = f(y,t)dt + g(y,t)dW \tag{31}$$

Discretize time and integrate over a short time interval $\Delta t$

$$\int_t^{t+\Delta t} dy = \int_t^{t+\Delta t} f(y,t')dt' + \int_t^{t+\Delta t} g(y,t')dW(t')$$

Using a left-endpoint rule with only a single interval one obtains the Euler-Maruyama scheme

$$y_{n+1} = y_n + f(y_n,t_n)\,\Delta t + g(y_n,t_n)\,\Delta W_n \tag{32}$$

where

$$\Delta W_n = W(t_n + \Delta t) - W(t_n).$$

The $\Delta W_n$ are Gaussian distributed with variance $\Delta t$. They are $\delta-$correlated

$$\langle \Delta W_n\, \Delta W_{n'} \rangle = \Delta t\, \delta_{n,n'}.$$

Using a normally distributed variable $\Delta\tilde{W}$ that has variance 1 we get $\Delta W_n$ by setting

$$\Delta W_n = \sqrt{\Delta t}\,\Delta\tilde{W} \qquad \text{with} \qquad P(\Delta\tilde{W}) = \frac{1}{\sqrt{2\pi}} e^{-\frac{\Delta\tilde{W}^2}{2}} \tag{33}$$

The noise strength is characterized by $g(y_n,t_n)$.

**Notes:**

- For each time step generate a new random number $\Delta W_n$ that obeys Gaussian statistics (normal distribution); in matlab this is done with `randn`.
- If y is a vector, usually the random processes for different components of y are independent: for each component of y one has to generate a different independent random number

To check the convergence of the strong approximation we need to compare solutions with different time steps $\Delta t$ for the **same realization** $W(t)$. A good strategy is to first generate the increments for a Wiener process (Brownian walk) with the smallest $\Delta t$ needed and then coarsen that walk by factors of 2 by adding adjacent increments.

1. Start with smallest $\Delta t$, generate increments $\Delta W_n^{(0)}$ for all time steps $t_n$ using a random number generator for a normal (Gaussian) distribution with variance $\Delta t$ (cf. (24,33)).

2. Increase the time step to $\Delta t^{(1)} \equiv 2\Delta t$ and generate the corresponding Wiener process with increments $\Delta W_n^{(1)}$ by adding pairs of successive increments,

$$\Delta W_n^{(1)} = \Delta W_{2n}^{(0)} + \Delta W_{2n+1}^{(0)} \qquad n = 0, 1, ... \tag{34}$$

   Use this Wiener process to run the code with time step $2\Delta t$.

3. Continue to add up the appropriate increments of the Wiener process with increments $\Delta W_n^{(l)}$ to generate Wiener processes with increments $\Delta W_n^{(l+1)}$, $l = 2 \ldots$, corresponding to time steps $\Delta t^{(l+1)} = 2^{l+1}\Delta t$. These compound increments have the variance

$$\langle \left(\Delta W_n^{(l)}\right)^2 \rangle = 2^l$$

   since

$$\begin{aligned} \left\langle \left(\Delta W_1^{(2)}\right)^2 \right\rangle &= \left\langle \left(\Delta W_1^{(1)} + \Delta W_2^{(1)}\right)^2 \right\rangle \\ &= \left\langle \left(\Delta W_1^{(1)}\right)^2 \right\rangle + \underbrace{\left\langle \Delta W_1^{(1)} \Delta W_2^{(1)} \right\rangle}_{=0} + \left\langle \left(\Delta W_2^{(1)}\right)^2 \right\rangle \end{aligned}$$

4. Use $2^l\Delta t$ and $\Delta W_n^{(l)}$ , $l = 1, 2, \ldots$, for successively less accurate approximations in (32).

**Note:** for $f(y_n, t_n) = 0$ and $g(y_n, t_n) = 1$ the Euler scheme (32) generates exactly the same $W(t)$ for all $l$. Changing the time step for a given realization makes no difference since the coarsened Brownian motion adds the steps of the finer representation of the realization.

**Order:**

- One can show that, in general, the forward Euler scheme has a global truncation error of $\mathcal{O}(\Delta t^{\frac{1}{2}})$.
- For $\frac{dg}{dy} = 0$, i.e. for additive noise, forward Euler is of order $\mathcal{O}(\Delta t^1)$ as in the deterministic case (see (36) below)

**Demo:**

- Demo1: wiener_coarse , coarsening of the Wiener process ntjjump=1 njmaax=4 nstepp=13 amp=0 field=0 g1=0 g0=1
- Demo2: Brown without potential: tmax=10 dt=0.1 nconf=100 then nconf=10000 to see how mean goes down as \sqrt{N}
- Demo3: Brown with quadratic potential: amp=0.1 nconf=1000 dt=0.1 tmax=100. variance grows initially and then saturates.
- Demo4: brown.conv with no potential and quadratic potential. with additive and multiplicative noise $nstepp = 8$ $ntjmax = 7$ $t_{max} = 5$ $amp = 0.5$ $field = 0.1$ $nconf = 100$ $x(1) = 0.5$.

### ii) Milstein Scheme

We want to get higher-order approximations.
Consider first again deterministic case

$$\frac{dy}{dt} = F(y)$$

**Note:**

- for simplicity we assume $F$ does not depend explicitly on time.

Rewrite as integral equation

$$y(t + \Delta t) = y(t) + \int_t^{t+\Delta t} F(y(t'))dt'$$

To evaluate the integral we rewrite the integrand

$$\begin{aligned} F(y(t')) &= F(y(t)) + \int_t^{t'} dF\left(y(t'')\right) \\ &= F\left(y(t)\right) + \int_t^{t'} \frac{\partial F}{\partial y}\frac{dy}{dt}dt = \\ &\underbrace{\approx}_{\text{left endpoint rule}} F\left(y(t)\right) + \left.\frac{\partial F}{\partial y}\right|_{y(t)} \left.\frac{dy}{dt}\right|_t (t' - t). \end{aligned}$$

Inserting this into the integral we get

$$\begin{aligned} y(t+\Delta t) &= y(t) + \Delta t F\left(y(t)\right) + \frac{\partial F}{\partial y}\frac{dy}{dt}\int_t^{t+\Delta t} t' - t\, dt' \\ &= y(t) + \Delta t F\left(y(t)\right) + \frac{1}{2}\Delta t^2 \frac{\partial F}{\partial y}\frac{dy}{dt} \end{aligned}$$

**Note:**

- This is the $2^{nd}$-order Taylor-series scheme we had early on.

Analogously, consider now the stochastic differential equation in integral form

$$y(t+\Delta t) = y(t) + \int_t^{t+\Delta t} F\left(y(t')\right)\, dt' + \int_t^{t+\Delta t} g\left(y(t')\right)\, dW(t')$$

We want to go beyond the left-end-point rule to evalute these integrals.

Again, we write the integrands as integrals over differentials

$$F\left(y(t')\right) = F\left(y(t)\right) + \int_t^{t'} dF\left(y(t'')\right)$$

Because $y$ is a stochastic process we have to use Ito's formula to calculate the differential $dF$ and analogously $dg$.

For any function $v(y)$ with $y$ satisfying the stochastic differential equation (28) Ito's formula gives

$$dv = \left(\frac{dv}{dy}F + \frac{1}{2}\frac{d^2v}{dy^2}g^2\right)\, dt + \frac{dv}{dy}g\, dW$$

Rewriting Ito's formula as an integral we get

$$v(y(t')) = v(y(t)) + \int_t^{t'}\left(\frac{dv}{dy}F + \frac{1}{2}\frac{d^2v}{dy^2}g^2\right)\, dt'' + \int_t^{t'}\frac{dv}{dy}g\, dW(t'')$$

Use this with $v(y) = F(y)$ and $v(y) = g(y)$ to rewrite the two integrands

$$\begin{aligned} y(t+\Delta t) &= y(t) + \qquad\qquad (35) \\ &\quad + \int_t^{t+\Delta t}\left[F(y(t)) + \int_t^{t'}\left(\frac{dF}{dy}F + \frac{1}{2}\frac{d^2F}{dy^2}g^2\right)\, dt'' + \int_t^{t'}\frac{dF}{dy}g\, dW(t'')\right] dt' \\ &\quad + \int_t^{t+\Delta t}\left[g(y(t)) + \int_t^{t'}\left(\frac{dg}{dy}F + \frac{1}{2}\frac{d^2g}{dy^2}g^2\right)\, dt'' + \int_t^{t'}\frac{dg}{dy}g\, dW(t'')\right] dW(t') \\ &= \underbrace{y(t) + \Delta t\, F(y(t)) + g(y(t))\,\Delta W}_{\text{Euler-Maruyama}} + h.o.t. \end{aligned}$$

We want the leading-order term beyond the Euler-Maruyama scheme: since $dW^2 = dt$ the dominant term is

$$\int_t^{t+\Delta t} \int_t^{t'} \frac{dg\left(y(t'')\right)}{dy} g\left(y(t'')\right)\, dW(t'')\, dW(t') \;=\; \frac{dg\left(y(t)\right)}{dy} g\left(y(t)\right) \int_t^{t+\Delta t} \int_t^{t'} dW(t'')\, dW(t') + \ldots$$

Here it was sufficient to evaluate the integrand at the left endpoint since this integral is part of the correction to the left endpoint approximation for the integral over $F(y(t'))$.

Evaluate the integral

$$\begin{aligned}
\int_t^{t+\Delta t} \int_t^{t'} dW(t'')\, dW(t') &= \int_t^{t+\Delta t} \left[W(t') - W(t)\right] dW(t') \\
&= \frac{1}{2}\left(W(t+\Delta t)^2 - W(t)^2 - \frac{1}{2}\Delta t\right) - W(t)\left(W(t+\Delta t) - W(t)\right) \\
&= \frac{1}{2}\left(W(t+\Delta t) - W(t)\right)^2 - \frac{1}{2}\Delta t = \frac{1}{2}\Delta W^2 - \frac{1}{2}\Delta t
\end{aligned}$$

**Note:**

- $\Delta W^2 \neq \Delta t$ in any given realization. Only $\langle \Delta W^2 \rangle = \Delta t$. The relation $dW^2 = dt$ between the differentials is also understood in the mean-square limit.

Omitting the other integral terms one obtains the Milstein scheme

$$y_{n+1} = y_n + \Delta t\, F(y_n) + g(y_n)\, \Delta W + \frac{1}{2}\frac{dg(y_n)}{dy} g(y_n)\left(\Delta W^2 - \Delta t\right) \tag{36}$$

**Notes:**

- As in (32) the Wiener process in (36) has variance $\Delta t$.
- Milstein scheme can be shown to have strong convergence of order $\Delta t^1$ (i.e. global truncation error).
- The additional effort to implement the Milstein scheme compared to the Euler-Maruyama scheme is small. It improves the accuracy significantly.
- For additive noise one has $\frac{dg}{dy} = 0$: the Euler-Maruyama scheme is then identical to the Milstein scheme and also becomes strong of $\mathcal{O}(\Delta t)$ (see homework)

To go beyond $\mathcal{O}(\Delta t)$ we need to deal with the other integrals

$$\int_t^{t+\Delta t} \int_t^{t'} dW(t'')dt' \qquad \int_t^{t+\Delta t} \int_t^{t'} dt''dW(t')$$

They cannot be expressed simply in terms of $\Delta W$ and $\Delta t$. We need to introduce an additional random variable

$$\Delta z = \int_t^{t+\Delta t} \int_t^{t'} dW(t'')dt'$$

One can show

$$\langle \Delta z \rangle = 0 \qquad \langle (\Delta z)^2 \rangle = \frac{1}{3}\Delta t^3 \qquad \langle \Delta z \Delta W \rangle = \frac{1}{2}\Delta t^2$$

**Note:**

- expect $\Delta z$ to be of $\mathcal{O}(\Delta t^{\frac{3}{2}})$, very loosely speaking.
- with $\Delta z$ included the scheme would become of $\mathcal{O}(\Delta t^{\frac{3}{2}})$.

### iii) Implicit Schemes

As in the deterministic case stability may severely limit the size of the time step and may require implicit schemes.

However, if one tries to implement a fully implicit scheme one runs into difficulties:

As an example, consider the backward Euler scheme for

$$dy = ay\,dt + by\,dW$$

$$y_{n+1} = \frac{y_n}{1 - a\Delta t - b\Delta W}$$

Since $\Delta W$ is Gaussian distributed and therefore unbounded and can have either sign the denominator can vanish (and change sign) for some $\Delta W$.

$\Rightarrow$ treat only the deterministic term implicitly

1. **Backward Euler**

$$y_{n+1} = y_n + \Delta t F(y_{n+1}, t_{n+1}) + g(y_n, t_n)\Delta W$$

2. **Backward Milstein**

$$y_{n+1} = y_n + \Delta t F(y_{n+1}, t_{n+1}) + g(y_n, t_n)\Delta W + \frac{1}{2}\frac{dg(y_n)}{dy}g(y_n)\left(\Delta W^2 - \Delta t\right)$$

### 11.2.2 Weak Approximation

If only averages and moments like $\langle y^n \rangle$ are of interest the strong approximation is not needed, i.e. any given run need not to converge to the exact solution corresponding to the given realization of the noise.

In particular:

The noise used in the simulation need not be the same noise as in the stochastic differential equation.

**i) Forward Euler**

For

$$y_{n+1} = y_n + F(y_n)\Delta t + g(y_n)\Delta\tilde{W}$$

the increments $\Delta\tilde{W}$ need not be Gaussian distributed. It is sufficient if $\Delta\tilde{W}$ satisfy the conditions

$$|\langle\Delta\tilde{W}\rangle| + |\langle\Delta\tilde{W}^3\rangle| + |\langle\Delta\tilde{W}^2\rangle - \Delta t| \leq K\Delta t^2 \tag{37}$$

**Notes:**

- The noise $\Delta\tilde{W}$ need not be a Wiener process, i.e. $\Delta\tilde{W}$ need not be $\int_t^{t+\Delta t} dW(t')$. Eq.(37) shows that the noise need only capture the first few moments of the Wiener process, which satisfies

  $$\langle\Delta W^{2m+1}\rangle = 0 \quad m \geq 0 \qquad \langle\Delta W^2\rangle = \Delta t$$

  Thus, the noise may differ from the Wiener process at order $\mathcal{O}(\Delta t^2)$.

- The noise could be a simple coin toss

  $$P(\Delta\tilde{W} = \pm\sqrt{\Delta t}) = \frac{1}{2}$$

  It seems however that in matlab there is no random number generator for such a dichotomic noise: one would have to use a Gaussian distributed number $x$ and then pick $\sqrt{\Delta t}$ or $-\sqrt{\Delta t}$ depending on the sign of $x$. This would be slower than using the Gaussian distribution.

- This weak Euler scheme has a weak convergence of $\mathcal{O}(\Delta t)$.

Demo: brown_converge.m:

- Euler and Milstein for $dy = -2Ay\,dt + (g0 + g1y)dW$ $A = 0.5$ with $y(0) = 1$. 1000 realizations nstepp=12 ntjmax=11 tmax=2 amp=0.5 field=0 g0=0,1 g1=1,0.

- Also without force and additive noise to show that even for large $dt$ the Wiener process itself is represented correctly, but that solving the ode induces error

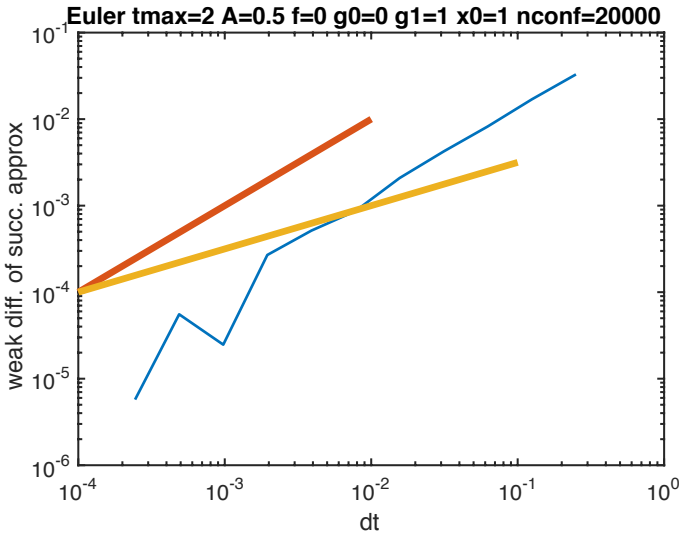



Figure 18: Weak error via difference between average of successive approximations with Euler-Maruyama scheme. $dy = -y\,dt + y\,dW$. Error $\propto \Delta t$. (The data should be marked with symbols as well).

### ii) Order-2 Weak Taylor Scheme

By keeping all the integrals in (35) and keeping also a term coming from $dW^2 dt$ at the next order one gets

$$\begin{aligned} y_{n+1} &= y_n + F_n\Delta t + g_n\Delta W + \frac{1}{2}g_n\frac{dg_n}{dy}\left(\Delta W^2 - \Delta t\right) + \\ &+ \frac{dF_n}{dy}g_n\Delta z + \frac{1}{2}\left(F_n\frac{dF_n}{dy} + \frac{1}{2}\frac{d^2F_n}{dy^2}g_n^2\right)\Delta t^2 + \\ &+ \left(F_n\frac{dg_n}{dy} + \frac{1}{2}\frac{d^2g_n}{dy^2}g_n^2\right)\left(\Delta W\Delta t - \Delta z\right) \end{aligned}$$

with

$$\Delta z = \int_t^{t+\Delta t}\int_t^{t'} dW(t'')dt'$$

For weak convergence $\Delta W$ can be replaced by $\Delta\tilde{W}$ and $\Delta z$ by $\frac{1}{2}\Delta\tilde{W}\Delta t$ if

$$|\langle\Delta\tilde{W}\rangle| + |\langle\Delta\tilde{W}^3\rangle| + |\langle\Delta\tilde{W}^5\rangle| + |\langle\Delta\tilde{W}^2\rangle - \Delta t| + |\langle\Delta\tilde{W}^4\rangle - 3\Delta t^2| \leq K\Delta t^3$$

**Notes:**

- the conditions are satisfied by a Gaussian random variable and also by a three-state discrete random variable with

$$P(\Delta\tilde{W} = \pm\sqrt{3\Delta t}) = \frac{1}{6} \qquad P(\Delta\tilde{W} = 0) = \frac{2}{3}$$

With that replacement we get the simplified weak scheme

$$\begin{aligned} y_{n+1} &= y_n + F_n\Delta t + g_n\Delta\tilde{W} + \frac{1}{2}g_n\frac{dg_n}{dy}\left(\Delta\tilde{W}^2 - \Delta t\right) + \frac{1}{2}\left(F_n\frac{dF_n}{dy} + \frac{1}{2}\frac{d^2F_n}{dy^2}g_n^2\right)\Delta t^2 \\ &\quad + \frac{1}{2}\left(\frac{dF_n}{dy}g_n + F_n\frac{dg_n}{dy} + \frac{1}{2}\frac{d^2g_n}{dy^2}g_n^2\right)\Delta\tilde{W}\Delta t \end{aligned}$$

**Note:**

- To generate $\Delta\tilde{W}$ generate a uniformly distributed variable $\xi \in [0,1]$ (using `rand` in matlab) and pick $\pm\sqrt{\Delta t}$ or $0$ depending on the value of $\xi$,

$$\begin{aligned} 0 \le \xi \le \frac{1}{6} \qquad \Delta\tilde{W} &= +\sqrt{3\Delta t} \\ \frac{1}{6} < \xi < \frac{1}{3} \qquad \Delta\tilde{W} &= -\sqrt{3\Delta t} \\ \frac{1}{3} < \xi \le 1 \qquad \Delta\tilde{W} &= 0 \end{aligned}$$

### 11.2.3 Application of Weak Approximation: Feynman-Kac Formula

**Idea:**

- heat diffuses
- Brownian motion underlies particle diffusion
- $\Rightarrow$ use Brownian motion (Wiener process) to describe heat diffusion

Consider the heat equation for a bar that is cooled by ambient **air**

$$\frac{\partial T}{\partial t} = \frac{1}{2}\frac{\partial^2 T}{\partial x^2} - K(x)T$$

with initial condition

$$T(x,0) = T_0(x)$$

**Note:**

- use Newton's law of cooling with $x-$dependent coefficient (e.g. $x-$dependent circulation)

Consider first solution without the cooling term,

$$T(x,t) = \int_{-\infty}^{\infty} \frac{1}{\sqrt{2\pi t}} e^{-\frac{(x-x')^2}{2t}} T_0(x')\,dx' \tag{38}$$

Check by plugging in.

**Note:** We can state this fact in the form

$$T(x,t) = \int_{-\infty}^{\infty} G(x,t;x',0)\, T_0(x')\, dx'$$

with $G(x,x')$ being the Green's function for the heat equation

$$\frac{\partial G(x.t;x',0)}{\partial t} = \frac{1}{2}\frac{\partial^2 G(x,t;x',0)}{\partial x^2} \qquad \lim_{t\to 0} G(x,t;x',0) = \delta(x-x')$$

$$G(x,t;x',0) = \frac{1}{\sqrt{2\pi t}} e^{-\frac{(x-x')^2}{2t}}$$

The diffusion equation can be solved using a weak approximation to the stochastic differential equation for random walker:

Probability distribution for increments of Wiener process

$$P(\Delta W) = \frac{1}{\sqrt{2\pi\Delta t}} e^{-\frac{\Delta W^2}{2\Delta t}}$$

By the central limit theorem we have that for large $n$ the sum over all the steps is Gaussian distributed with

$$\left\langle \sum_{j=1}^{n} \Delta W_j \right\rangle = n \left\langle \Delta W \right\rangle = 0 \qquad \left\langle \left( \sum_{j=1}^{n} \Delta W_j \right)^2 \right\rangle = n \left\langle (\Delta W)^2 \right\rangle = n\, \Delta t$$

we get the probability distribution function for the position $W(t) = x + \sum_{j=1}^{n} \Delta W_j$ at time $t = n\Delta t$ of a walker that starts at $x$ at time $t = 0$,

$$P_x(W,t) = \frac{1}{\sqrt{2\pi n\Delta t}} e^{-\frac{(W-x)^2}{2n\Delta t}} = \frac{1}{\sqrt{2\pi t}} e^{-\frac{(W-x)^2}{2t}}$$

i.e. the variance grows linearly with the number of steps.

Thus, one can read the solution (38) as

$$T(x,t) = \int_{-\infty}^{\infty} P_x(W,t) T_0(W)\, dW \equiv \langle T_0(W(t)) \rangle_x$$

**Notes:**

- here $\langle ... \rangle_x$ denotes the average over Wiener processes that start at position $x$
- to get the temperature at a position $x$ and time $t$ one starts many random walkers at position $x$, which at time $t$ would reach $W(t)$. The temperature is given by the value of the average of $T_0(W(t))$ over those walks.
- the picture of diffusing random walkers would have suggested the converse procedure:
  start with $T_0(x')$ random walkers at position $x'$ and time $t = 0$. At time $t$ their distribution would give the temperature distribution.

- since $P_x(W,t) = P_W(x,t)$ the two approaches are equivalent
- Starting with walkers at $x'$ would not allow to compute the temperature at a given position $x$ directly. Would have to average over walks starting at many different initial positions, most of which would not end up at $x$ and would therefore not contribute to the value of $T(x,t)$.

With cooling the walkers (corresponding to 'heat particles') at position $W(t)$ 'decay' at a rate $K(t, W(t))$
Over a given path $W(t)$ the number of walkers 'shrinks' by a factor

$$e^{-\int_0^t K(W(t'))dt'}$$

therefore one gets the Feynman-Kac formula

$$T(x,t) \;=\; \langle e^{-\int_0^t K(W(t'))dt'} T_0(W(t))\rangle_x \equiv \int_{-\infty}^{\infty} P_x(W,t)\, e^{-\int_0^t K(W(t'))dt'} T_0(W)\, dW$$

**Note:**

- again, it does not matter whether the walker goes from $x$ to $W(t)$ or the other way round; the decay is the same.

**Note:**

- Feynman introduced this formula in the context of the Schrödinger equation

$$i\frac{\partial \psi}{\partial t} = \frac{\partial^2 \psi}{\partial x^2} + V(x)\psi$$

for the complex wave function $\psi(x,t)$,which determines the probability to find a quantum-mechanical particle at position $x$.

To get the expectation value solve the problem

$$\begin{aligned} dy &= dW \qquad y(0) = x \\ du &= K(t, y(t))\, dt \qquad u(0) = 0 \end{aligned}$$

then

$$T(x,t) = \langle e^{-\int_0^t K(t', W(t'))dt'} T_0(W(t))\rangle_x = \langle e^{-u(t)} T_0(y(t))\rangle_x$$

**Note:**

- Through the initial condition $y(0) = x$ the average is over random walks that start at position $x$
- Since we are only interested in mean values here, we can replace $dW$ by $d\tilde{W}$

$$\begin{aligned} y_{n+1} &= y_n + \Delta\tilde{W} \qquad \text{with} \qquad P(\Delta W = \pm\sqrt{\Delta t}) = \frac{1}{2} \\ u_{n+1} &= u_n + K(y_n)\,\Delta t \end{aligned}$$

# 12 Fluctuations and Tipping Points

As an application of stochastic differential equations we consider systems that undergo sudden transitions to quite different states when a parameter is changed, i.e. they exhibit saddle-node bifurcations or first-order transitions.

Simple mechanical example: catastrophic snap-through of a pre-stressed beam.

- In this transition the system goes to a state that is very different from the initial state.
- This system has strong hysteresis: once the beam has snapped through it is not sufficient to release the load to get it back into the original position.

Such catastrophic transitions have also been discussed in the context of climate change. The worry is that through the increasing amount of greenhouse gases in the atmosphere, which reduce the amount of infrared radiation that transports heat away from the earth, could lead to a rise in the earth's temperature that triggers a transition to a climate that would be so different from our current climate that adaptation would be very difficult (e.g. significantly rising ocean levels).

Small Ice-Cap Instability:

Ice reflects sun light more strongly than water. Therefore, the melting of ice (Greenland, arctic, antarctica) increases the amount of radiation absorbed by the earth which increases the energy influx. This could lead to further melting of ice. This positive feedback may lead to a run-away melting. Conversely, if an ice cap becomes large enough it could reduce the temperature sufficiently to lead to further growth of the ice.

Consider a minimal energy-balance model for the small ice-cap instability[18] .

$$c\frac{dT}{dt} = (1 - \alpha(T))\, F_{sw} - \epsilon\sigma T^4$$

Here

- $F_{sw}$ the influx of energy through short-wave radiation
- $\sigma T^4$ the black-body radiation emitted by the earth (Stefan-Boltzmann law).
- $\alpha(T)$ is the reduction of the influx by albedo, which is assumed to depend on the temperature via the temperature-dependence of the coverage of the earth

[18] D. Notz, *The future of ice sheets and sea ice: between reversible retreat and unstoppable loss*, PNAS 106 (2009) 20590.

with ice.
A reasonable fit is apparently obtained with[19]

$$\alpha(T) = \begin{cases} \alpha_i & T \leq T_i \\ \alpha_o + (\alpha_i - \alpha_o)\frac{(T-T_o)^2}{(T_i-T_o)^2} & T_i \leq T < T_o \\ \alpha_o & T \geq T_o \end{cases}$$

with $\alpha_i > \alpha_o$. This function is meant to mimic that for $T > T_o$ all of the oceans are free of ice, while for $T < T_i$ they are completely covered with ice (snowball earth). The steeper slope for lower temperature captures that there is more area near the equator where the ice melts first when a snowball earth is heated up and therefore a change in temperature has a bigger effect on the albedo. Conversely, there is very little area near the poles and a change in temperature has not much impact on the albedo once almost all ice has melted.
$T$ is an average temperature of that model earth $\Rightarrow$ to have all oceans be frozen over $T$ has to be well below the freezing temperature of water since the temperature is much higher at the equator and conversely for the transition to the ice-free state.

- $\epsilon$ is an effective emissivity describing the partial opaqueness of the atmosphere to long-wave (thermal) radiation. This emissivity is reduced by greenhouse gases.

Fixed points and bifurcation diagram:

Solve for $\epsilon\sigma$

$$\epsilon\sigma = \frac{F_{sw}}{T^4}\left[1 - \alpha(T)\right]$$

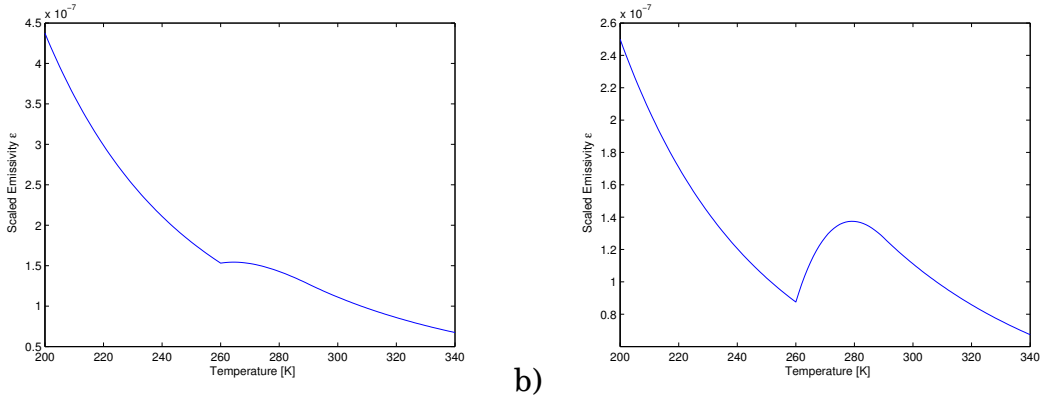



Figure 19: Fixed points for the energy-balance model for $T_i = 260$, $T_o = 290$, $\alpha_o = 0.1$. a) $\alpha_i = 0.3$ b) $\alpha_i = 0.6$.

To read Fig.19 as a bifurcation diagram, flip it over. The dynamics is the identified by considering limiting values of the temperature: For $T \to \infty$ one gets $dT/dt$<0.

[19] R. Pierrehumbert, *Principles of Planetary Climate*, 2009.

- For high values of the effective emissivity there is a single solution: snowball earth ($T < T_i$)
- For small values of the effective emissivity the only solution is ice-free earth ($T > T_o$)
- For intermediate values three solutions exist, two of them have partial ice covering.

With a change in emissivity the saddle-node bifurcations can be passed and a sudden transition occurs. In this model one has two possible jumps

- decreasing $\epsilon$ one has a jump from a snowball earth to either ice-free or a partially frozen earth
- increasing $\epsilon$ one has a jump from a partially frozen earth to a snowball earth.

Near the bifurcation points small perturbations of the temperature could induce a jump transition: dangerous!

Can one predict the vicinity of a bifurcation point before it is too late?

Consider a yet simpler model with a simpler nonlinearity

$$\frac{dy}{dt} = F(y) \equiv \mu y + \beta y^3 + \gamma y^5 + h \tag{39}$$

with $\mu < 0$, $\beta > 0$ and $\gamma < 0$.

This system has a potential, i.e. its dynamics can be written as

$$\frac{dy}{dt} = -\frac{dU}{du} \qquad \text{with } U = -\frac{1}{2}\mu y^2 - \frac{1}{4}\beta y^4 - \frac{1}{6}\gamma y^6 - fy.$$

As $\mu$ is increased from very negative values a second minimum develops for $y > 0$ and eventually the minimum near $y = 0$ disappears in a saddle-node bifurcation.As this point is approached the minimum near $y = 0$ becomes quite shallow since it merges with a maximum. Mathematically, the saddle-node bifurcation is determined by the condition

$$\left.\frac{dF}{dy}\right|_{\text{at fixed point}} = 0 \qquad \Leftrightarrow \qquad \frac{d^2U}{dy^2} = 0 \quad \text{and} \quad \frac{dU}{dy} = 0.$$

Even small noise can induce large flucutations in such a shallow minimum

- consider the variance of $y$ as a function of $\mu$ and test whether one can indeed discern a substantial increase as $\mu$ is ramped towards the saddle-node bifurcation.

One goal of the homework is to assess what conditions are more favorable to use such fluctuations as early warning signs in (39).

# 13 Two-Point Boundary-Value Problems

Consider a different type of ordinary differential equation: boundary value problems

**Examples:**

1. Placing a satellite
   equation of motion for satellite

$$m\frac{d^2x}{dt^2} = F(x,t)$$

   usually one has initial values

$$x(0) = x_0 \qquad \left.\frac{dx}{dt}\right|_{t=0} = v_0$$

   When placing a satellite task is different. Given are

$$x(0) = x_0 \qquad x(T) = x_{final}$$

   initial velocity is not given; even final time may not be prescribed.

2. Eigenvalue problems

   (a) Consider the motion of a string

$$\frac{\partial^2 u}{\partial t^2} = c(x)^2\frac{\partial^2 u}{\partial x^2} \qquad u(0,t) = 0 \qquad u(L,t) = 0$$

   where the wave speed $c(x)$ may depend on the position, because the thickness of the string varies.
   Look for harmonic motion

$$u(x,t) = e^{i\omega t}U(x)$$

$$c(x)^2\,\frac{d^2U(x)}{dx^2} = -\omega^2 U(x)$$

   eigenmodes and eigenfrequencies are determined by this boundary-value problem:
   we need to choose $\omega$ such that $U(x)$ satisfies both boundary conditions

   (b) Heat diffusion

$$\frac{\partial T}{\partial t} = \frac{\partial}{\partial x}\left(D(x)\frac{\partial T}{\partial x}\right) \qquad \alpha_{0,L}T + \beta_{0,L}\,\frac{\partial T}{\partial x} = \gamma_{0,L} \quad \text{at } x = 0, L$$

   again linear and time-independent coefficients: exponential ansatz

$$T(x,t) = e^{\sigma t}\Phi(x)$$

$$\frac{\partial}{\partial x}\left(D(x)\frac{\partial \Phi}{\partial x}\right) = \sigma\,\Phi \qquad \alpha_{0,L}\Phi + \beta_{0,L}\,\frac{\partial \Phi}{\partial x} = \gamma_{0,L} \quad \text{at } x = 0, L$$

or interested in steady temperature distribution

$$\frac{\partial}{\partial x}\left(D(x)\frac{\partial T(x)}{\partial x}\right) = 0 \qquad \alpha_{0,L}T + \beta_{0,L}\,\frac{\partial T}{\partial x} = \gamma_{0,L}$$

3. Electrostatics
   Poisson equation for the potential $\Phi$ generated by density $\rho$ of free charges

$$\nabla \cdot (\epsilon \mathbf{E}) = 4\pi\rho \qquad \mathbf{E} = -\nabla\Phi$$

   with $\epsilon$ the dielectric coefficient of the material.
   Typically the potential is given on the boundary
   In one dimension

$$\frac{\partial}{\partial x}\left(\epsilon(x)\frac{\partial \Phi}{\partial x}\right) = 4\pi\rho(x) \qquad \Phi(0) = \Phi_0 \quad \Phi(L) = \Phi_L$$

   where dielectric coefficient can depend on location.

## 13.1 Shooting Method

Consider first an analytical example: mass-spring system

$$\frac{d^2x}{dt^2} + \frac{k}{m}x = 0 \tag{40}$$

to get a unique solution we usually pose initial conditions

$$x(0) = x_0 \qquad \left.\frac{dx}{dt}\right|_{t=0} = v_0$$

Consider here the boundary-value problem

$$x(0) = x_0 \qquad x(t_f) = x_f$$

The general solution for the differential equation (40) is given by

$$x(t) = A\cos(\omega t) + B\sin(\omega t) \qquad \text{with} \quad \omega^2 = \frac{k}{m}$$

We need

$$x(0) = x_0 = A \qquad x(t_f) = x_f = A\cos(\omega t_f) + B\sin(\omega t_f)$$

We need to tune $v_0$ such that the final position is the desired position. Since we have the general *analytical* solution, we have an explicit expression that connects

the initial velocity with the final position: we can simply solve for $B$, which happens to be proportional to the initial velocity $v_0 = \omega B$,

$$B = \frac{x_f - x_0 \cos(\omega t_f)}{\sin(\omega t_f)}$$

Thus

$$x(t) = x_0 \, \cos \omega t + \frac{x_f - x_0 \cos(\omega t_f)}{\sin(\omega t_f)} \, \sin \omega t$$

In the numerical method we also have to tune $v_0$, but we cannot simply solve for it. But we have a numerical way to connect $v_0$ to the final position. We then 'fiddle' with $v_0$ until we hit the desired final position.

To stick to our general formulation in terms of first-order systems, we rewrite the second-order equations as two first-order equations

$$\begin{aligned} \frac{dx}{dt} &= v \\ \frac{dv}{dt} &= \omega^2 x \end{aligned}$$

implemented, e.g., as forward Euler

$$\begin{aligned} x_{n+1} &= x_n + \Delta t \, v_n \\ v_{n+1} &= v_n + \Delta t \, \omega^2 x_n \end{aligned}$$

To even get started we need to have $x_0$ ***and*** $v_0$.
We have to guess $v_0$ and see where this leads to in terms of $x(t_f)$
Write this as a formal map that takes the initial condition with yet unknown $v_0$ and maps it to the final state

$$x(t_f) = X_f(x_0, v_0).$$

Then we need to solve

$$G(v_0) \equiv X_f(x_0, v_0) - x_f = 0$$

i.e. we need need to find a root of $G(u)$: use Newton iteration

$$u^{(l+1)} = u^{(l)} - \frac{1}{\frac{dG}{du}\Big|_{u^{(l)}}} G(u^{(l)})$$

We need the derivative $\frac{dG}{du}$: approximate it by comparing $G$ for closely spaced initial conditions for $v$, $u^{(l)}$ and $u^{(l)} + \delta u$,

$$\left.\frac{dG}{du}\right|_{u^{(l)}} \approx \frac{G(u^{(l)} + \delta u) - G(u^{(l)})}{\delta u}$$

**Note:**

- evaluating $G(u)$ requires solving the ordinary differential equation over the whole time interval; this can be a slow process: If $u$ is a scalar, one can use $u^{(l)}$ and $u^{(l+1)}$ as comparisons for the derivative to avoid solving the ODE twice in each Newton step for successive steps

$$\frac{dG}{du} \approx \frac{G(u^{(l)}) - G(u^{(l-1)})}{u^{(l)} - u^{(l-1)}}$$

  This will work only if the steps $u^{(l)} - u^{(l-1)}$ are not too large. It will not work if multiple initial conditions need to be determined by the Newton iteration, i.e. if $u$ is a vector.

**Algorithm**

1. guess $u^{(1)} \equiv v_0^{(1)}$
2. use the initial velocity $u^{(l)}$ to solve the ODE numerically to get $X_f(x_0, u^{(l)})$
3. perform a Newton iteration step

$$u^{(l+1)} = u^{(l)} - \frac{1}{\frac{X_f(x_0, u^{(l)}+\delta u) - X_f(x_0, u^{(l)})}{\delta u}} \left(X_f(x_0, u^{(l)}) - x_f\right)$$

   to get the Jacobian (derivative of $G$) the ODE is also solved using a nearby initial condition for the velocity $u^{(l)} + \delta u$
4. go to step 2 using $u^{(l+1)}$ as the updated initial velocity, unless $X_f(x_0, u^{(l+1)})$ is close enough to $x_f$, i.e. $\left|G(u^{(l+1)})\right|$ is small enough.
5. upon convergence $u \to u^{(\infty)}$: $v_0 = u^{(\infty)}$ and

$$x(t_f) = X_f(x_0, u^{(\infty)})$$

## 13.2 Application: Optimal Control of a Mass in a Potential

A more powerful approach to direct a mass towards a goal is to vary not only the initial condition but to apply a controling force all along the trajectory, e.g. igniting boosters on the rocket transporting the satellite.

Consider the two-dimensional motion of a mass in some potential with possibly some friction controled by an externally time-dependent force

$$m\frac{d^2\boldsymbol{x}}{dt^2} = -\nabla U(\boldsymbol{x}) - \beta\frac{d\boldsymbol{x}}{dt} + \boldsymbol{f}(t).$$

Write the equation as a first-order system

$$\dot{x} = v \tag{41}$$
$$\dot{y} = w \tag{42}$$
$$\dot{v} = -\frac{1}{m}\partial_x U(x,y) - \frac{\beta}{m}v + \frac{1}{m}f_x \tag{43}$$
$$\dot{w} = -\frac{1}{m}\partial_y U(x,y) - \frac{\beta}{m}w + \frac{1}{m}f_y \tag{44}$$

The goal is to direct the mass from some initial state

$$[x(0), y(0), v(0), w(0)]$$

to a specific final state

$$[x(t_f), y(t_f), v(t_f), w(t_f)]$$

in a fixed time $t_f$.

Now there are infinitely many different ways to get the mass into the target. To make the problem meaningful, we introduce a *cost function*

$$P = \int_0^{t_f} E(\boldsymbol{f}(t), \boldsymbol{x}(t), \frac{d}{dt}\boldsymbol{x}(t))\, dt.$$

Now the goal is to find a solution that goes from the initial position to the final position and *minimizes* the cost function.

A reasonable cost function is

$$P = \frac{1}{2}\int_0^{t_f} \boldsymbol{f}(t)^2\, dt$$

Why not take the work performed on the mass as the cost function? Not only accelerating, but also decelerating takes effort, even though work is extracted from the mass. With this in mind one could consider a more general cost function

$$P = \alpha\,\frac{1}{2}\int_0^{t_f} \boldsymbol{f}(t)^2\, dt + (1-\alpha)\,\frac{1}{2}\int_0^{t_f} (f_x v)^2 + (f_y w)^2\; dt.$$

Again, since performing work on the mass as well as extracting work from the mass is associated with effort. Therefore, we do not want the $x$-component of the work to be able to balance the $y$-dependent component, which would be the case if we took $\int (\mathbf{f}\cdot\mathbf{v})^2\, dt$. The parameter $\alpha$ allows to weigh the two components differently. It turns out that the control method we will use does not work very well for small $\alpha$.

## 13.3 Minimization with Constraints: Pontryagin's Principle

Consider first the somewhat simpler problem of obtaining the optimal path for a mass moving in 1 dimension with a single control variable. For that we seek

- the minimum of a cost function

$$P = \int_o^{t_f} E\left(x(t'), f(t')\right) dt'$$

- under the constraint

$$\frac{d}{dt}x(t) = F(x(t), f(t))$$

- where $f(t)$ is a time-dependent control variable/force that should enable us to 'steer' the solution $x(t)$ to the desired final position $x(t_f) = x_f$.

**Note:**

- Here we have allowed the cost function to depend also on $x(t)$.

Consider first a yet simpler example:
Find the minimum $\mathrm{x}_m$ of $P(\mathrm{x})$ under the constraint $g(\mathrm{x}) = 0$ with $\mathrm{x} \in \mathbb{R}^n$

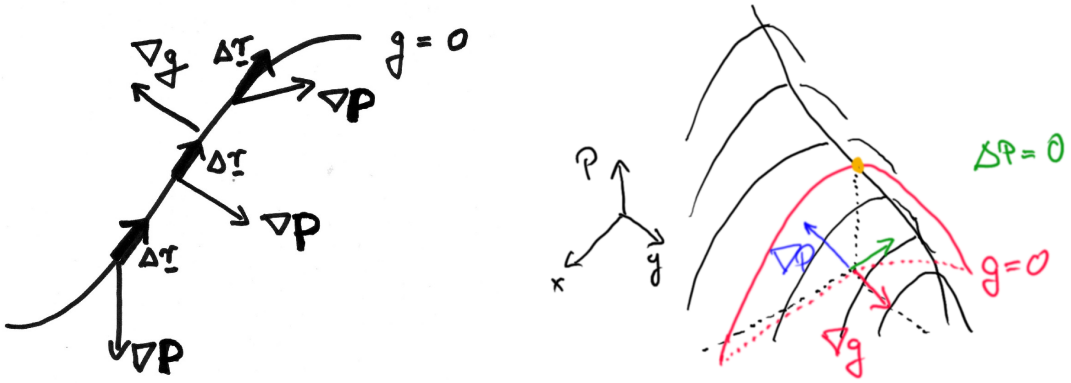



Figure 20: The condition for an extremum under a constraint is given by $\nabla P \parallel \nabla g$.

Consider the change of $P(\boldsymbol{x})$ for an infinitesimal change in $\boldsymbol{x}$

$$\Delta \mathrm{P} = \nabla \mathrm{P} \cdot \Delta \mathbf{x}$$

We are looking for points at which $\Delta P = 0$.

Compare the minimum under a constraint with a regular local minimum:

Local Minimum without constraint:

- At a local minimum $\mathbf{x}_m$ the function $P(\mathbf{x})$ does not change to *linear order* when $\mathbf{x}$ is shifted by an amount $\Delta\mathbf{x}$ in *any direction* $\Rightarrow \nabla P(\boldsymbol{x}_m) = 0$.

Minimum under constraint:

- At a minimum $\mathbf{x}_m$ under the *constraint* $g(\mathbf{x}) = 0$ the function $P(x)$ does not change *to linear order* if $\mathbf{x}$ is changed by an amount $\Delta\mathbf{x}$ along the surface defined by $g(\mathbf{x}) = 0$.

Rephrase the condition for the constrained minimum:

- At a minimum $\mathbf{x}_m$ under a constraint the gradient $\nabla P$ need not be zero, but $P(\mathbf{x})$ is only allowed to change in directions $\Delta\mathbf{x}$ that violate the constraint $g(\mathbf{x}) = 0$, i.e. in the direction in which $g(\mathbf{x})$ changes: $\nabla g(\mathbf{x})$.

Thus

$$\nabla P \| \nabla g \qquad \Rightarrow \qquad \nabla P + \lambda \nabla g = 0$$

**Note:**

- the parameter $\lambda$ is called a Lagrange multiplier

In general, for multiple constraints $g_k(\mathbf{x}) = 0$ and higher dimensions:
$\nabla P$ need not be 0, it can have components in the directions in which one of the $g_k$ changes, because changes of $\mathbf{x}$ in those directions are not allowed by the corresponding constraint,

$$\nabla P = \tilde{\lambda}_1 \nabla g_1 + \tilde{\lambda}_2 \nabla g_2 + ... \tilde{\lambda}_K \nabla g_K$$

Write it as

$$\nabla \left( P(\mathbf{x}) + \sum_{k=1}^{K} \lambda_k \, g_k(\mathbf{x}) \right) = 0 \tag{45}$$

**Now:** How do we deal with an *integral* and the constraint given by a *differential equation*? The differential equation represents infinitely many constraints, one for each time point $t$.
Consider a finite-difference discretization of the integral and the differential equation using $x_n = x(t_n)$ and $f_n = f(t_n)$, $n = 1 \ldots N$:

We seek

- a minimum of

$$P = \Delta t \sum_{n=1}^{N} E(x_n, f_n)$$

- under $N$ constraints corresponding to the $N$ discrete time points $t_n$

$$\frac{(x_{n+1}-x_n)}{\Delta t} = F(x_n, f_n) \qquad 1 \le n \le N$$

which are combined to

$$\Delta t \sum_{n=1}^{N} \lambda_n \, g_n(\mathbf{x}, \mathbf{f}) \qquad \text{with} \qquad g_n(\boldsymbol{x}, \boldsymbol{f}) = -\frac{x_{n+1}-x_n}{\Delta t} + F(x_n, f_n)$$

**Note:**

- The vector x in (45) is now replaced by $x_n$ together with $f_n$:

$$\mathbf{x} = (x_1, ...x_N, f_1, ...f_N)$$

- $\nabla$ is also replaced,

$$\nabla \;\; \rightarrow \;\; (\frac{\partial}{\partial x_1}, ..., \frac{\partial}{\partial x_N}, \frac{\partial}{\partial f_1}, ..., \frac{\partial}{\partial f_N})$$

The Lagrange multiplier equation (45) reads now

$$\frac{\partial}{\partial x_j}\left\{\Delta t \sum_{n=1}^{N} E(x_n, f_n) + \sum_{n=1}^{N} \lambda_n \, (x_n + \Delta t \, F(x_n, f_n) - x_{n+1})\right\} = 0 \tag{46}$$

and

$$\frac{\partial}{\partial f_j}\left\{\Delta t \sum_{n=1}^{N} E(x_n, f_n) + \sum_{n=1}^{N} \lambda_n \, (x_n + \Delta t \, F(x_n, f_n) - x_{n+1})\right\} = 0 \tag{47}$$

Using $\frac{\partial x_n}{\partial x_j} = \delta_{jn}$ and $\frac{\partial f_n}{\partial f_j} = \delta_{jn}$ one gets

$$\Delta t \, \frac{\partial E(x_j, f_j)}{\partial x_j} + \lambda_j \left(1 + \Delta t \frac{\partial F(x_j, f_j)}{\partial x_j}\right) - \lambda_{j-1} = 0$$

Thus

$$\lambda_{j-1} = \lambda_j + \Delta t \, \left(\lambda_j \frac{\partial F(x_j, f_j)}{\partial x_j} + \frac{\partial E(x_j, f_j)}{\partial x_j}\right)$$

Here

$$\lambda_j = \lambda(t_j) \qquad \lambda_{j-1} = \lambda(t_j - \Delta t).$$

Replacing $\Delta t \rightarrow -\Delta t$ we see that this expression corresponds to the forward Euler scheme for

$$\boxed{\frac{d\lambda(t)}{dt} = -\lambda \, \frac{\partial F(x, f)}{\partial x} - \frac{\partial E(x, f)}{\partial x},} \tag{48}$$

which is to be solved in parallel to

$$\boxed{\frac{dx(t)}{dt} = F(x(t), f(t)).} \tag{49}$$

The connection between $u(t)$ and $\lambda(t)$ is given by (47) which results in

$$\Delta t \frac{\partial E}{\partial f_j} + \lambda_j \Delta t \frac{\partial F}{\partial f_j} = 0,$$

i.e. the algebraic equation

$$\boxed{\lambda \frac{\partial F(x,f)}{\partial f} + \frac{\partial E(x,f)}{\partial f} = 0.} \tag{50}$$

**Notes:**

- For a single dynamical variable $x$(t) we have a single constraint and a single Lagrange parameter $\lambda(t)$, each of which satisfies a differential equation.
- We have an algebraic equation connecting the Lagrange parameter $\lambda(t)$ with the control variable $f(t)$. It is an algebraic equation rather than a differential equation because $f$ appears only with the index $n$ and not $n \pm 1$, i.e. no derivative of $f$ appears in the equation.
- Both differential equations are initial-value problems, i.e. we need to specify $x(0)$ and $\lambda(0)$.
  For our control problem we are only given $x(0)$. The initial value $\lambda(0)$ is arbitrary, but we have to satisfy $x(t_f) = x_f$. Thus, the condition for the final value $x(t_f)$ needs to be translated into an initial condition $\lambda(0)$.

Apply now the results from the single dynamical variable to the equations for the two-dimensional motion.

- The dynamical system is given by four variables $(x(t), y(t), v(t), w(t))$. Thus, there are 4 variables $x^{(i)}(t)$ satisfying the 4 differential equations (41,42,43,44), which imply 4 constraints at any given time $t$. To satisfy these 4 constraints $g^{(k)}$ we need four Lagrange parameters $\lambda^{(i)}(t)$ at any given time.

$$\dot{x} = v \qquad \Rightarrow \qquad \lambda^{(1)}(t) \tag{51}$$

$$\dot{y} = w \qquad \Rightarrow \qquad \lambda^{(2)}(t) \tag{52}$$

$$\dot{v} = -\frac{1}{m}\partial_x U(x,y) - \frac{\beta}{m} v + \frac{1}{m} f_x \qquad \Rightarrow \qquad \lambda^{(3)}(t) \tag{53}$$

$$\dot{w} = -\frac{1}{m}\partial_y U(x,y) - \frac{\beta}{m} w + \frac{1}{m} f_y \qquad \Rightarrow \qquad \lambda^{(4)}(t) \tag{54}$$

  Write the dynamical system as

$$\frac{dx^{(k)}}{dt} = F^{(k)}(x^{(j)}, f^{(j)}) \qquad k = 1 \ldots 4.$$

Using discrete time steps $t_n$, $n = 1 \ldots N$, as in (46), eq.(45) becomes then

$$\nabla \left( \sum_{n=1}^{N} E + \sum_{k=1}^{4} \sum_{n=1}^{N} \lambda_n^{(k)} g_n^{(k)} \right) = 0.$$

Here

$$\nabla = \left( \frac{\partial}{\partial x_1^{(1)}}, \ldots, \frac{\partial}{\partial x_n^{(k)}}, \ldots \frac{\partial}{\partial x_N^{(4)}}, \frac{\partial}{\partial f_{x1}}, \ldots, \frac{\partial}{\partial f_{yN}} \right).$$

Thus, the $4N$ constraints $g_n^{(k)}$ defining the dynamical system are added together.

- Comparison with (46) and (48) shows that in continuous time the Lagrange multipliers $\lambda^{(k)}(t)$ satisfy the evolution equations

$$\begin{aligned} \frac{d\lambda^{(i)}}{dt} &= -\frac{\partial E(x^{(1)}, x^{(2)}, x^{(3)}, x^{(4)}, f_x, f_y)}{\partial x^{(i)}} \\ &\quad - \sum_{k=1}^{4} \lambda^{(k)} \frac{\partial F^{(k)}(x^{(1)}, x^{(2)}, x^{(3)}, x^{(4)}, f_x, f_y)}{\partial x^{(i)}} \qquad i = 1 \ldots 4 \end{aligned} \tag{55}$$

- There are two control variables

$$f_x(t) \qquad f_y(t)$$

They are related to the Lagrange multipliers via (50)

$$\frac{\partial E(f_x, f_y)}{\partial f_x} + \sum_{k=1}^{4} \lambda^{(k)} \frac{\partial F_k(x^{(j)}, f_x, f_y)}{\partial f_x} = 0 \tag{56}$$

$$\frac{\partial E(f_x, f_y)}{\partial f_y} + \sum_{k=1}^{4} \lambda^{(k)} \frac{\partial F_k(x^{(j)}, f_x, f_y)}{\partial f_y} = 0 \tag{57}$$

Evaluating (55,56,57) we obtain

$$\frac{d\lambda^{(1)}}{dt} = \frac{1}{m} \lambda^{(3)} \partial_{xx} U + \frac{1}{m} \lambda^{(4)} \partial_{xy} U \tag{58}$$

$$\frac{d\lambda^{(2)}}{dt} = \frac{1}{m} \lambda^{(3)} \partial_{xy} U + \frac{1}{m} \lambda^{(4)} \partial_{yy} U \tag{59}$$

$$\frac{d\lambda^{(3)}}{dt} = -\lambda^{(1)} + \lambda^{(3)} \frac{\beta}{m} - (1 - \alpha) f_x^2 v \tag{60}$$

$$\frac{d\lambda^{(4)}}{dt} = -\lambda^{(2)} + \lambda^{(4)} \frac{\beta}{m} - (1 - \alpha) f_y^2 w \tag{61}$$

with

$$f_x = -\frac{1}{m \left[ \alpha + (1 - \alpha) v^2 \right]} \lambda^{(3)} \qquad f_y = -\frac{1}{m \left[ \alpha + (1 - \alpha) w^2 \right]} \lambda^{(4)} \tag{62}$$

**Now:**

- Known are the *initial* and *final* conditions for $x$, $y$, $v$, $w$
- Unknown are the initial conditions for $\lambda^{(i)}$, $i = 1...4$
- $\Rightarrow$ total count is o.k.: 8 equations for 8 variables

Now the situation is similar to that of positioning a satellite:
Some of the initial conditions have been replaced by final conditions.

**Shooting method:**

Consider the map $\mathbf{X}_{t_f}$ from the initial conditions $\mathbf{x}_0 = (x_0, y_0, v_0, w_0)$ and $\boldsymbol{\lambda_0} = (\lambda_0^{(1)}, \lambda_0^{(2)}, \lambda_0^{(3)}, \lambda_0^{(4)})$ to the state of the system at time $t_f$,

$$\mathbf{x}(t_f) = \mathbf{X}_{t_f}(\mathbf{x}_0, \lambda_\mathbf{0})$$

For a given $\mathbf{x}_0$ we need to satisfy a 4-component vector function

$$\mathbf{X}_{t_f}(\mathbf{x}_0, \lambda_\mathbf{0}) = \mathbf{x}_f \qquad \Leftrightarrow \qquad \mathbf{G}(\lambda_\mathbf{0}) \equiv \mathbf{X}_{t_f}(\mathbf{x}_0, \lambda_\mathbf{0}) - \mathbf{x}_f = 0.$$

To obtain $\boldsymbol{\lambda_0}$ we use the Newton iteration

$$\boldsymbol{u}^{(l+1)} = \boldsymbol{u}^{(l)} - \boldsymbol{J}^{-1}\boldsymbol{G}(\boldsymbol{u}^{(l)})$$

which is solved as

$$\boldsymbol{J}(\boldsymbol{u}^{(l)}) \left(\boldsymbol{u}^{(l+1)} - \boldsymbol{u}^{(l)}\right) = -\boldsymbol{G}(\boldsymbol{u}^{(l)})$$

with $\boldsymbol{J}(\boldsymbol{u}^{(l)})$ the Jacobian of G

$$J_{ij} = \frac{\partial G^{(i)}}{\partial \lambda_0^{(j)}}.$$

$\mathrm{G}(\boldsymbol{u}^{(l)})$ is obtained by solving (51-54) and (58-61) numerically using a suitable time-stepping routine with the initial condition $\boldsymbol{\lambda_0}$ given by $\boldsymbol{u}^{(l)}$.
$\boldsymbol{J}$ is obtained by solving (51-54) and (58-61) with slightly modified initial conditions for one $\lambda_0^{(i)}$ at a time

$$\frac{\partial G^{(i)}}{\partial \lambda^{(j)}} \approx \frac{G^{(i)}(\lambda_0^{(1)}, ..\lambda_0^{(j)} + \delta\lambda, ...\lambda_0^{(4)}) - G^{(i)}(\lambda_0^{(1)}, ..., \lambda_0^{(4)})}{\delta\lambda} \qquad \delta\lambda \ll 1$$

Since $\mathbf{x}(t_f)$ is the relevant output, we continue the Newton iteration until $|\boldsymbol{G}\left(\lambda_0\right)| = |\mathbf{x}(t_f) - \mathbf{x}_f| < tol$
Upon convergence set $\boldsymbol{\lambda_0} = \boldsymbol{u}^\infty$, which gives then the correct initial conditions for $\boldsymbol{\lambda}$
The time-dependence of the required forces is then given by the time-dependence of $\boldsymbol{\lambda}(t)$ via (62).

**Components of Code:**

- Main program
  - calls Newton with an initial guess for $\lambda$
  - using the output from the converged Newton call the time evolution to obtain the correct path $[x(t), y(t), v(t), w(t)]$.
- Newton solver
  - call time evolution to get G
  - call Jacobian
  - iterate over initial conditions for $\lambda$
  - upon convergence output the initial conditions for $\lambda_i$ that yield the correct final conditions $[x(t_f), y(t_f), v(t_f), w(t_f)]$
- Jacobian
  - call time evolution to get $J_{ij}$
- time evolution
  - call time-stepping routine with the initial conditions for $\lambda$ as input
  - can be any kind depending on task:
    * order of scheme
    * adaptive/non-adaptive
    * explicit vs. implicit

**Note:**

- The convergence of the Newton iteration always depends strongly on the initial guess. In the implicit time-stepping methods the solution at the previous time step is usually a good initial guess. In the current problem there is no simple way to guess a good initial set of $\lambda$. A good strategy is then: pick a guess randomly and instead of iterating for a very large number of steps and hoping for the best cancel the iteration after a fixed number of steps and pick a new random guess

### 13.3.1 Pontryagin's Principle using Functional Derivatives

The derivation of (48) and (50) can be done without introducing finite-difference approximations. Use *variational calculus* instead.

In order to write (46) without introducing discrete times introduce

$$W\left\{x(t), f(t)\right\} \equiv \int_0^{t_f} E\left(x(t'), f(t')\right) + \lambda(t')\left(-\frac{dx}{dt} + F(x(t'), f(t'))\right) dt'$$

**Note:**

- $W\{x(t)\}$ is called a *functional* of $x(t)$: it maps a function into a scalar number
- one can think of $x(t)$ as the $t^{th}-$component of the infinite-dimensional vector $x$
- $W$ is then a function of infinitely many vector components $x(t)$

Introduce the *functional derivative* $\frac{\delta W}{\delta x(t)}$ instead of the partial derivative $\frac{\partial W}{\partial x_j}$ via

$$\delta W \equiv W\{x(t)+\delta x(t)\} - W\{x(t)\} = \frac{\delta W}{\delta x(t)}\delta x(t)$$

where the function $\delta x(t)$ is *arbitrary* except that it has to vanish at the end-points $t=0$ and $t=t_f$: $x(0)=x_0$ and $x(t_f)=x_f$

$$\delta x(0) = 0 = \delta t(t_f) \tag{63}$$

We will omit $f(t)$ for now for simplicity

$$\delta W = \int_0^{t_f} E\left(x(t)+\delta x(t)\right) + \lambda(t)\left\{-\frac{d}{dt}\left(x(t)+\delta x(t)\right) + F\left(x(t)+\delta x(t)\right)\right\} dt - W\{x(t)\}$$

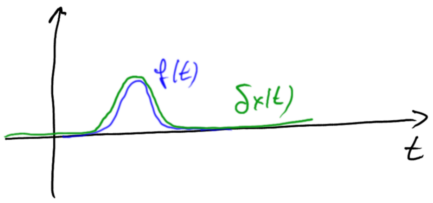
expanding the integrand in $\delta x$ cancels the terms without $\delta x$

$$\delta W = \int_0^{t_f} \frac{\partial E}{\partial x}\delta x + \lambda\left\{-\frac{d}{dt}\left(\delta x\right) + \frac{\partial F}{\partial x}\delta x\right\} dt$$

This expression is complicated because it contains $\delta x(t)$ as well as $\frac{d}{dt}\delta x(t)$. Perform integration by parts; because of (63) the boundary terms vanish,

$$\delta W = \int_0^{t_f} \frac{\partial E}{\partial x}\delta x + \frac{d}{dt}\left(\lambda\right)\delta x(t) + \lambda\frac{\partial F}{\partial x}\delta x(t)dt = \int_0^{t_f}\left(\frac{\partial E}{\partial x} + \frac{d}{dt}\left(\lambda\right) + \lambda\frac{\partial F}{\partial x}\right)\delta x(t)dt$$

We are looking for a saddle of $W$, i.e. a 'point' $x(t)$ for which $W\{x(t)\}$ does not change when $x(t)$ is slightly variedIn order for $\delta W = 0$ for arbitrary $\delta x(t)$ the term in the parenthesis has to vanish




$$\frac{\partial E}{\partial x} + \frac{d}{dt}\lambda + \lambda\frac{\partial F}{\partial x} = 0$$

Recover the dependence on $f(t)$, which appears *without* a time-derivative in $W$: no integration by parts needed and therefore and no time-derivative of $\lambda$ arises

$$\frac{\partial E}{\partial f} + \lambda \frac{\partial F}{\partial f} = 0$$

When there are $K$ dependent variables and $M$ control functions

$$\begin{aligned} W\left\{x_1(t), x_2(t), ..., x_K(t), f_1(t), ...f_M(t)\right\} &= \int_0^{t_f} E(..., x_i(t'), ..., f_j(t')...)dt' + \\ &+ \sum_{k=1}^{K} \lambda_k(t') \left( -\frac{dx_k}{dt} + F_k(..., x_i, ...f_j...) \right) dt' \end{aligned}$$

one obtains

$$\begin{aligned} \frac{d}{dt}\lambda_i &= -\frac{\partial E}{\partial x_i} - \sum_{k=1}^{K} \lambda_k \frac{\partial F_k}{\partial x_i} \\ 0 &= -\frac{\partial E}{\partial f_j} - \sum_{k=1}^{K} \lambda_k \frac{\partial F_k}{\partial f_j} \end{aligned}$$

## 13.4 Shooting Method for Linear Problems

For linear differential equations one has *superposition principle*
$\Rightarrow$ can determine coefficients for the linear superposition directly without Newton iteration

Revisit as a simple example the harmonic oscillator with arbitrary time-dependent inhomogeneous terms

$$\frac{d}{dt}\begin{pmatrix} x \\ v \end{pmatrix} - \begin{pmatrix} 0 & v \\ \omega^2 & 0 \end{pmatrix} \begin{pmatrix} x \\ v \end{pmatrix} = \begin{pmatrix} I_x(t) \\ I_v(t) \end{pmatrix} \tag{64}$$

with general boundary conditions

$$\mathbf{B}_0 \begin{pmatrix} x(0) \\ v(0) \end{pmatrix} + \mathbf{B}_f \begin{pmatrix} x(t_f) \\ v(t_f) \end{pmatrix} = \mathbf{b}$$

**Note:**

- Here $I_x(t)$ and $I_y(t)$ are fixed time-dependent functions; they are not control functions.
- The boundary conditions $x(0) = x_0$, $x(t_f) = x_f$ correspond then to

$$\mathbf{B}_0 = \begin{pmatrix} 1 & 0 \\ 0 & 0 \end{pmatrix} \qquad \mathbf{B}_f = \begin{pmatrix} 0 & 0 \\ 1 & 0 \end{pmatrix} \qquad \mathbf{b} = \begin{pmatrix} x_0 \\ x_f \end{pmatrix}$$

Since the equations are linear and inhomogeneous equation, the general solution is given by a superposition of the general solution for the homogeneous problem and a particular solution

$$\begin{pmatrix} x(t) \\ v(t) \end{pmatrix} = s_1 \begin{pmatrix} x_1(t) \\ v_1(t) \end{pmatrix} + s_2 \begin{pmatrix} x_2(t) \\ v_2(t) \end{pmatrix} + \begin{pmatrix} x_p(t) \\ v_p(t) \end{pmatrix}$$

**Note:**

- it does not really matter much which initial condition the particular solution satisfies, since the initial (and terminal condition) will be taken care of by the general homogeneous solution.

which can be written as

$$\mathbf{y}(t) = \mathbf{Y}(t)\,\mathbf{s} + \mathbf{y}_p(t)$$

with the fundamental solution $\mathbf{Y}(t)$ given by

$$\mathbf{Y}(t) = \begin{pmatrix} x_1(t) & x_2(t) \\ v_1(t) & v_2(t) \end{pmatrix}$$

and

$$\mathbf{s} = \begin{pmatrix} s_1 \\ s_2 \end{pmatrix} \qquad \mathbf{y}_p(t) = \begin{pmatrix} x_p(t) \\ v_p(t) \end{pmatrix}$$

The fundamental solution satisfies the initial conditions

$$\mathbf{Y}(0) = \mathbf{I} \equiv \begin{pmatrix} 1 & 0 \\ 0 & 1 \end{pmatrix}$$

The boundary condition can then be written as

$$\mathbf{B}_0 \left(\mathbf{Y}(0)\mathbf{s} + \mathbf{y}_p(0)\right) + \mathbf{B}_f \left(\mathbf{Y}(t_f)\mathbf{s} + \mathbf{y}_p(t_f)\right) = \mathbf{b}$$

$$\underbrace{\left(\mathbf{B}_0\mathbf{Y}(0) + \mathbf{B}_f\mathbf{Y}(t_f)\right)}_{\mathbf{M}}\,\mathbf{s} + \underbrace{\mathbf{B}_0\mathbf{y}_p(0) + \mathbf{B}_f\mathbf{y}_p(t_f) - \mathbf{b}}_{-\hat{\mathbf{b}}} = 0$$

Obtain the coefficients s then by

$$\mathbf{s} = \mathbf{M}^{-1}\hat{\mathbf{b}}$$

**Notes:**

- no Newton iteration needed to determine coefficients s**.**
- both $M$ and $\hat{b}$ require a solution at $t_f$, which in general will be obtained numerically

**Algorithm for $n$ coupled equations:**

1. integrate *homogeneous* initial-value problem $n$ times with $n$ initial conditions

$$\begin{pmatrix} 1 \\ 0 \\ ... \\ 0 \end{pmatrix}, \begin{pmatrix} 0 \\ 1 \\ 0 \\ ... \end{pmatrix}, ... \begin{pmatrix} 0 \\ ... \\ 0 \\ 1 \end{pmatrix} \longleftrightarrow \mathbf{Y}(0)$$

2. integrate *inhomogeneous* initial-value problem once with initial condition

$$\mathbf{y}_p(0) = \begin{pmatrix} 0 \\ 0 \\ 0 \\ 0 \end{pmatrix}$$

3. determine s from

$$\mathbf{s} = \mathbf{M}^{-1}\hat{\mathbf{b}}$$

4. since $\boldsymbol{y}_p(0)$ was chosen to vanish the initial conditions for the solution are given by

$$\mathbf{y}(0) = \mathbf{Y}(0)\mathbf{s} + \boldsymbol{y}_p(0) = \mathbf{s}$$

   Solve the equations with that initial condition.

**Notes:**

- last step needed:
  - the fundamental solution typically does not get stored at all intermediate times
  - for different initial conditions adaptive time step may have led to different intermediate times in the different columns of $\mathbf{Y}(t)$
  - solution may be needed at different intermediate times than were used for $\mathbf{Y}(t)$
- problems can arise if the fundamental solution contains exponentially growing solutions
  round-off errors can lead to large errors in particular when solving the matrix equation for s
- as usually, solve for s as $\mathrm{s} = \mathrm{M}\backslash\hat{\mathrm{b}}$
- typical application: eigenvalue problems, since they are linear

## 13.5 Finite-Difference Method

Consider the boundary-value problem

$$\frac{d^2y}{dx^2} = F(y, \frac{dy}{dx}, x)$$

with two different types of boundary conditions

$$\begin{aligned} y(0) &= g_0 \\ \alpha_L y(L) + \beta_L \frac{dy}{dx}(L) &= g_L \end{aligned}$$

Use finite differences to approximate the derivatives

$$\begin{aligned} y''(x_n) &= \frac{1}{\Delta x^2}\left(y_{n+1} - 2y_n + y_{n-1}\right) + \mathcal{O}\left(\Delta x^2\right) \\ y'(x_n) &= \frac{1}{2\Delta x}\left(y_{n+1} - y_{n-1}\right) + \mathcal{O}\left(\Delta x^2\right) \end{aligned}$$

As for the diffusion equation, the two types of boundary conditions are treated differently

- $x = 0$: the value of $y$ is given directly at the boundary $\Rightarrow$ the boundary point is not determined by the differential equation.

$$\frac{1}{\Delta x^2}\left(y_2 - 2\underbrace{y_0}_{g_0} + y_1\right) = F\left(y_1, \frac{1}{\Delta x}\left(y_2 - \underbrace{y_0}_{g_0}\right), x_1\right)$$

- at $x = L$: $y_N$ is not given directly by the boundary condition, we need to introduce a fictitious point $x_{N+1}$ and determine $y_{N+1}$ from the boundary condition

$$\alpha_L y_N + \beta_L\left(\frac{y_{N+1} - y_{N-1}}{2\Delta x}\right) = g_L$$

leading to

$$y_{N+1} = y_{N+1}\left(y_{N-1}, y_N\right) = y_{N-1} + \frac{2\Delta x}{\beta_L}\left(g_L - \alpha_L y_N\right)$$

and

$$\frac{1}{\Delta x^2}\left(\underbrace{y_{N+1}}_{\text{from b.c.}} - 2y_N + y_{N-1}\right) = F\left(y_N, \frac{1}{\Delta x}\left(\underbrace{y_{N+1}}_{\text{from b.c.}} - y_{N-1}\right), x_N\right)$$

We then get

$$\frac{1}{\Delta x^2}\begin{pmatrix} -2 & 1 & 0 & \dots & \\ 1 & -2 & 1 & 0 & \\ 0 & 1 & -2 & 1 & 0 \\ \dots & & 1 & -2 & 1 \\ & & 0 & 1+1 & -2-\frac{2\Delta x}{\beta_L}\alpha_L \end{pmatrix}\begin{pmatrix} y_1 \\ y_2 \\ y_i \\ y_{N-1} \\ y_N \end{pmatrix} =$$

$$= \begin{pmatrix} F\left(y_1, \frac{1}{\Delta x}\left(y_2 - g_0\right), x_1\right) \\ \dots \\ F\left(y_i, \frac{1}{\Delta x}\left(y_{i+1} - y_{i-1}\right), x_i\right) \\ \dots \\ F\left(y_N, \frac{1}{\Delta x}\left(\frac{2\Delta x}{\beta_L}\left(g_L - \alpha_L y_N\right)\right), x_N\right) \end{pmatrix} +$$

$$+ \begin{pmatrix} -\frac{1}{\Delta x^2}\, g_0 \\ 0 \\ \dots \\ 0 \\ -\frac{1}{\Delta x^2}\,\frac{2\Delta x}{\beta_L}\, g_L \end{pmatrix}$$

**Notes:**

- If $F$ is linear
  - one has to solve this system only once, with $F$ modifying the matrix on the left-hand side.
  - the system is tridiagonal, therefore relatively efficient to solve: operation count is linear in $N$

Use Newton's method to solve the nonlinear system

$$\mathbf{G}\left(\mathbf{y}\right) \equiv \mathbf{D}\mathbf{y} - \mathbf{F}\left(\mathbf{y}\right) - \mathbf{g} = 0$$

$$\mathbf{y}^{(m+1)} = \mathbf{y}^{(m)} - \mathbf{J}\left(\mathbf{y}^{(m)}\right)^{-1}\left(\mathbf{D}\mathbf{y}^{(m)} - \mathbf{F}\left(\mathbf{y}^{(m)}\right) - \mathbf{g}\right)$$

with the Jacobian

$$\mathbf{J} = \mathbf{D} - \underbrace{d\mathbf{F}\left(\mathbf{y}^{(m)}\right)}_{\text{Jacobian of } \mathbf{F}}$$

$d\mathbf{F}$ is also tridiagonal

$$\begin{aligned}
\frac{\partial F_1}{\partial y_j} &= \frac{\partial F_1(u,v,x)}{\partial u}\delta_{j,1} + \frac{1}{\Delta x}\frac{\partial F_1(u,v,x)}{\partial v}\delta_{j,2} \\
\frac{\partial F_i}{\partial y_j} &= \frac{\partial F_i(u,v,x)}{\partial u}\delta_{j,i} + \frac{1}{\Delta x}\frac{\partial F_i(u,v,x)}{\partial v}\left(\delta_{j,i+1} - \delta_{j,i-1}\right) \qquad 1 < i < N \\
\frac{\partial F_N}{\partial y_j} &= \frac{\partial F_N(u,v,x)}{\partial u}\delta_{j,N} + \frac{1}{\Delta x}\frac{\partial F_N(u,v,x)}{\partial v}\left(-\frac{2\alpha_L}{\beta_L}\right)\delta_{j,N} \qquad i = N
\end{aligned}$$

Therefore the Newton iterations are also very efficient, solving only the tridiagonal linear system

$$\mathbf{J}\left(\mathbf{y}^{(m)}\right)\left(\mathbf{y}^{(m+1)}-\mathbf{y}^{(m)}\right)=-\mathbf{D}\mathbf{y}^{(m)}+\mathbf{F}\left(\mathbf{y}^{(m)}\right)+\mathbf{g}$$

Since the tridiagonal matrix $\mathbf{J}$ is sparse, it is important to make use of that in MATLAB:

```
v=ones(N,1);
D=-2*diag(v)+diag(v(1:N-1),1)+diag(v(1:N-1),-1);
Ds=sparse(D);
```

Computation time for solving the equation $\mathbf{Dx}=\mathbf{b}$ 10,000 times using `x=D\b` and for $\mathbf{D}*\mathbf{D}$

| $N$ | non-sparse solve | sparse solve | non-sparse multiply | sparse multiply |
|---|---|---|---|---|
| 5 | 0.067 | 0.013 | 0.011 | 0.026 |
| 10 | 0.096 | 0.014 | 0.015 | 0.026 |
| 100 | 1.65 | 0.036 | 0.61 | 0.077 |
| 200 | 7.98 | 0.060 | 3.90 | 0.13 |
| 400 | 33.7 | 0.11 | 28.6 | 0.29 |

**Notes:**

- The extremely high speed of the sparse solver is presumably due it using the Thomas algorithm for the tri-diagonal matrix, which scales like $N$ rather than $N^2$.
- To get higher-order accuracy one can increase the order of the finite-difference approximation

  $$\frac{d^2y}{dx^2}=\frac{1}{12}\frac{1}{\Delta x^2}\left(-y_{n+2}+16y_{n+1}-30y_n+16y_{n_1-1}-y_{n-2}\right)+\mathcal{O}\left(\Delta x^4\right)$$

  this increases the stencil and the width of the differentiation matrix
- Adaptive grid spacing: how to control?

# 14 Differential-Algebraic Equations

Consider the set of stiff equations

$$\begin{aligned}\frac{d}{dt}\mathbf{y} &= \mathbf{F}(\mathbf{y},\mathbf{z})\\ \frac{d}{dt}\mathbf{z} &= \frac{1}{\epsilon}\mathbf{G}(\mathbf{y},\mathbf{z}) \qquad \epsilon\ll 1\end{aligned}$$

In the asymptotic limit $\epsilon \to 0$ it becomes

$$\begin{aligned} \frac{d}{dt}\mathbf{y} &= \mathbf{F}(\mathbf{y}, \mathbf{z}) \\ 0 &= \mathbf{G}(\mathbf{y}, \mathbf{z}) \qquad \epsilon \ll 1 \end{aligned}$$

Such eqution are called differential-algebraic equations, with y being the differential variables and z the algebraic ones.

The algebraic equations represent constraints on the dynamics and can emerge more generally than through such an asymptotic limit of dynamical equations.

Are there alternative methods to highly stable fully implicit methods?

## 14.1 Example: Pendulum in Cartesian Coordinates

Write the pendulum in cartesian coordinates

$$\begin{aligned} \frac{dx}{dt} &= u \\ \frac{dy}{dt} &= v \\ m\frac{du}{dt} &= -\frac{N(t)\,x}{\sqrt{x^2+y^2}} \\ m\frac{dv}{dt} &= -\frac{N(t)\,y}{\sqrt{x^2+y^2}} - mg \end{aligned}$$

with the constraint

$$x^2 + y^2 = L^2$$

Why not in polar coordinates? For more general constraints there may not be a coordinate transformation to incorporate the constraint

$$y = \frac{x^2}{1+x^2}$$

or general 'roller coaster'.

**Note:**

- The tension $N$ in the rod is not constant in time
  - when the pendulum hangs down the rod pulls the mass up
  - when the pendulum is up-side down the rod pushes the mass up
- Without knowledge of $N(t)$ the equations of motion cannot be solved
- Solving the constraint for $y$ has problems

- multi-valued
- does not directly provide information about $N(t)$

To simplify further analysis rewrite the equations using a rescaled tension

$$\eta = \frac{N(t)}{m\sqrt{x^2+y^2}}$$

to get

$$\begin{aligned} \frac{dx}{dt} &= u \\ \frac{dy}{dt} &= v \\ \frac{du}{dt} &= -\eta x \\ \frac{dv}{dt} &= -\eta y - g \\ x^2 + y^2 - L^2 &= 0 \end{aligned}$$

**Can we get an evolution equation for $\eta$?**

Take multiple derivatives of the constraint equation:

First derivative

$$xu + yv = 0$$

i.e. the velocity vector has to be orthogonal to the position vector to ensure the circular motion.

Second derivative

$$u^2 + x\dot{u} + v^2 + y\dot{v} = 0$$

$$u^2 + v^2 - \eta \underbrace{\left(x^2+y^2\right)}_{L^2} - gy = 0$$

This equation allows to solve for $\eta$

$$\eta = \frac{1}{L^2}\left(u^2 + v^2 - gy\right)$$

i.e. the tension in the rod has to balance the centrifugal force and gravity.

Third derivative

$$2u\dot{u} + 2v\dot{v} - \dot{\eta}L^2 - gv = -2\eta \underbrace{(ux+vy)}_{=0} - 2vg - \dot{\eta}L^2 - g\dot{v} = 0$$

Solving for $\dot{\eta}$

$$\dot{\eta} = -\frac{3gv}{L^2} \tag{65}$$

together with

$$\begin{aligned}
\frac{dx}{dt} &= u \\
\frac{dy}{dt} &= v \\
\frac{du}{dt} &= -\eta x \\
\frac{dv}{dt} &= -\eta y - g
\end{aligned}$$

The initial conditions are not arbitrary:

- they need to satisfy the constraint and its temporal derivatives, otherwise they immediately drive the system away from the constraint surface

$$\begin{aligned}
x(0)^2 + y(0)^2 &= L^2 \\
x(0)\,u(0) + y(0)\,v(0) &= 0 \\
\eta(0) &= \frac{1}{L^2}\left(u(0)^2 + v(0)^2 - g y(0)\right)
\end{aligned}$$

**Demo:** (dae.m)

- problem=2, tmax=100, theta=0.5pi, method=4, tol=0.01 and smaller: exponential growth of L

**Notes:**

- The scheme does not manage to keep the length of the pendulum exactly constant: the length shows drift, which in this case amounts to an exponential growth for larger $\Delta t$.
- The drift decreases with decreasing $\Delta t$.
- Since the growth of $L$ is exponential, it will be very difficult to run the simulations for long times, since ever smaller time steps will be needed.

**Notes:**

- The scheme does not enforce the constraint directly and exactly:
  - the scheme determines the change in the tension that is needed to keep the tension at the value that would be needed to keep the pendulum at the intended length.

  - if the numerical error has changed the length of the pendulum, there is no corrective action for the length and the tension is updated as if the length was still correct.

- For the pendulum we needed to take 3 time derivatives of the constraint to obtain a differential equation of the unknown variable $N(t)$: the system is said to have *index 3*.
- The higher the index the more indirect is connection between the constraint and the evolution equation that is derived from it.
- The numerical challenge increases with the index of the system.

## 14.2 Dealing with the Drift

### 14.2.1 Enforcing by Substitution

Consider a DAE with index 1. Its standard form is given by

$$\begin{aligned} \frac{d\mathbf{y}}{dt} &= \mathbf{F}(\mathbf{y}, \mathbf{z}, t) \\ 0 &= \mathbf{G}(\mathbf{y}, \mathbf{z}) \end{aligned}$$

where the Jacobian $\partial_{\mathbf{z}}\mathbf{G}$ is nonsingular.

Then the implicit function theorem guarantees a smooth solution $\mathbf{z}(\mathbf{y})$ that satisfies the constraint, i.e. one can solve the constraint equation for $\mathbf{z}$

Clearly taking 1 derivative gives an evolution equation for $\mathbf{z}$

$$\partial_{\mathbf{y}}\mathbf{G} \cdot \dot{\mathbf{y}} + \partial_{\mathbf{z}}\mathbf{G} \cdot \dot{\mathbf{z}} = 0$$

$$\dot{\mathbf{z}} = -\left(\partial_{\mathbf{z}}\mathbf{G}\right)^{-1} \partial_{\mathbf{y}}\mathbf{G} \cdot \dot{\mathbf{y}} = -\left(\partial_{\mathbf{z}}\mathbf{G}\right)^{-1} \cdot \mathbf{F}(\mathbf{y}, \mathbf{z}, t)$$

Since $\mathbf{G}$ can, in principle, be solved for $\mathbf{z}$ one can eliminate $\mathbf{z}$ to get

$$\frac{d\mathbf{y}}{dt} = \mathbf{F}(\mathbf{y}, \mathbf{z}(\mathbf{y}), t)$$

**Notes:**

- The equation for $\mathbf{z}$ may be nonlinear, making solving it difficult. Also, there may be multiple solutions; but as long as $\partial_{\mathbf{z}}\mathbf{G}$ is non-singular, ie. $\det\left(\partial_{\mathbf{z}}\mathbf{G}\right) \neq 0$ no solution branches merge or disappear.

For the pendulum:

Written as

$$\begin{aligned}
\frac{dx}{dt} &= u \\
\frac{dy}{dt} &= v \\
\frac{du}{dt} &= -\eta x \\
\frac{dv}{dt} &= -\eta y - g
\end{aligned}$$

with constraint

$$\eta = \frac{1}{L^2}\left(u^2 + v^2 - gy\right) \tag{66}$$

the pendulum equation is an index-1 DAE and it is easy to solve for $\eta$.

**Demo:** (dae.m)

- problem=1, tmax=60, ODE45, reltol=0.01 and 0.001, theta=0.3pi, 0.5pi, 0.55pi, 0.57pi, 0.7pi.
- compare with problem=2 for theta=0.5pi reltol=0.001,
- then reduce reltol=0.00001 theta=0.6pi P3 works P2 does not, worse for theta=0.9pi.

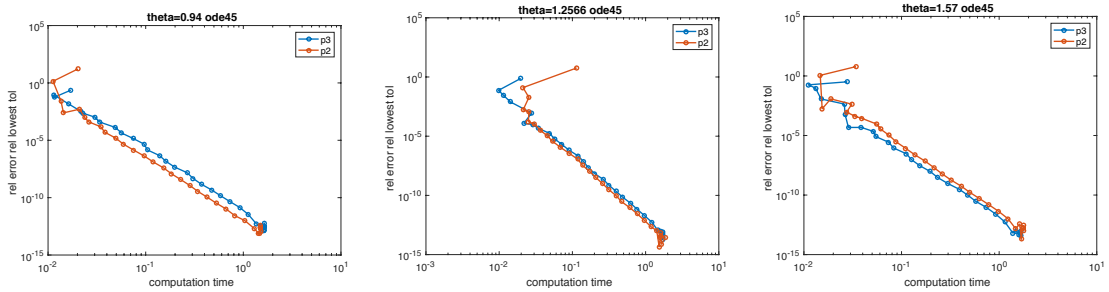



Figure 21: Comparison of solutions to pendulum equations with evolution equation for $\eta$ (65) (P3) and with inserting $\eta$ (66) (P2). For larger oscillation amplitudes inserting $\eta$ induces an instability that reduces its accuracy ($\theta_0 = 0.3\pi$, $\theta_0 = 0.4\pi$, $\theta_0 = 0.5\pi$; $t_{max} = 20$)

**Notes:**

- The constraint is much better enforced, no exponential growth
- The accuracy is much better for initial conditions that lead to small-amplitude oscillations

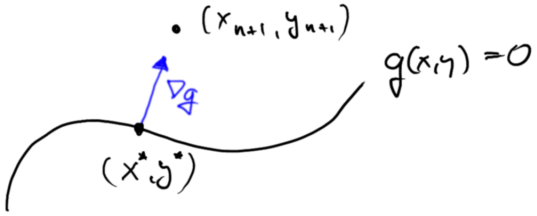
- For initial conditions closer to whirling motion, the scheme blows up and is unusable even for small $\Delta t$; in that regime the formulation with a differential equation for $\eta$ works better.

**Note:**

- For the substitution to be possible the DAE has to have index 1.

### 14.2.2 Brute-force constraint

In each time step one can force the solution back onto the constraint surface by a projection, i..e after having computed $\mathbf{y}^*_{n+1}$ from the differential equations one can find the point $\mathbf{y}_n$ on the constraint surface that is closest to $\mathbf{y}^*_{n+1}$.
For the pendulum one would modify $x$ and $y$ in the direction of $\nabla G(x,y) = \nabla\left(x^2+y^2\right)$,




Thus,

$$(x_{n+1}-x^*, y_{n+1}-y^*) = \alpha \nabla g(x,y) = \alpha(2x^*, 2y^*)$$

i.e.

$$(x^*, y^*) = \frac{1}{1+2\alpha}(x_{n+1}, y_{n+1})$$

Since

$$L^2 = x^{*2} + y^{*2} = \frac{1}{1+2\alpha}\left(x_{n+1}^2 + y_{n+1}^2\right)$$

and

$$1+2\alpha = \frac{\sqrt{x_{n+1}^2 + y_{n+1}^2}}{L}$$

we get

$$(x_{n+1}, y_{n+1}) = \left(\frac{L x^*_{n+1}}{\sqrt{x^{*2}_{n+1} + y^{*2}_{n+1}}}, \frac{L y^*_{n+1}}{\sqrt{x^{*2}_{n+1} + y^{*2}_{n+1}}}\right) \tag{67}$$

**But:**

- if the velocity $(u^*_{n+1}, v^*_{n+1})$ do not satisfy the constraint

  $$xu + yv = 0$$

  the solution would move away from the constraint $x^2 + y^2 = L^2$ right away again, because the motion is not tangential to the circle.

- if $\eta^*_{n+1}$ does not satisfy the balance of the centrifugal force

$$\eta = \frac{1}{L^2}\left(u^2 + v^2 - gy\right)$$

  the velocities would move away from tangential motion right away again.

- Thus, to avoid the drift one should
  - find the points $\mathbf{y}_{n+1}$ that are closest to $\mathbf{y}^*_{n+1}$
  - and that satisfy all the algebraic constraints that arise in the process of deriving the evolution equation for the constraint variable $\eta$

- Using Lagrange parameters one would look for critical points of

$$\left\|\mathbf{y}_{n+1} - \mathbf{y}^*_{n+1}\right\| + \left\|\mathbf{z}_{n+1} - \mathbf{z}^*_{n+1}\right\| + \lambda \cdot \tilde{\mathbf{G}}(\mathbf{y}, \mathbf{z})$$

  which can be quite a task.

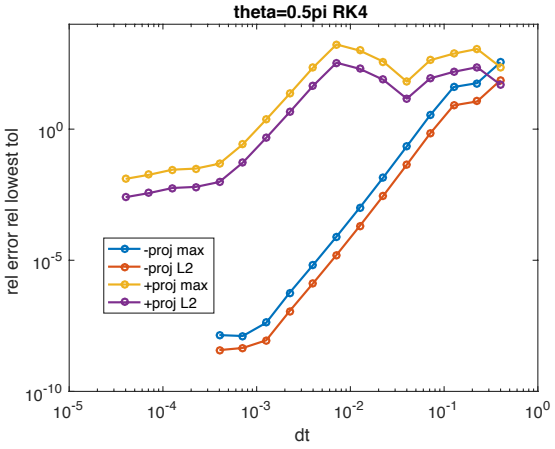



Figure 22: Brute-force enforcing only the length constraint for the pendulum does not improve overall accuracy even though length is fixed (-project: without projection, +project: with projection (67))

Demo

- enforcing only length: amplitude and frequency clearly affected and accuracy deteriorated, even though the length almost perfectly fixed: 093207 and 093916 tmax=100 P3 RK4 dt=0.4, 0.2 show significant change in amplitude and frequency.

### 14.2.3 Solve the Constraints together with the Differential Equations

Considering the equation with constraint of an index-1 DAE as the asymptotic limit of the stiff problem

$$\begin{aligned} \frac{d}{dt}\mathbf{y} &= \mathbf{F}(\mathbf{y},\mathbf{z}) \\ \frac{d}{dt}\mathbf{z} &= \frac{1}{\epsilon}\mathbf{G}(\mathbf{y},\mathbf{z}) \qquad \epsilon \ll 1 \end{aligned}$$

suggests using an unconditionally stable implicit method for the equations with contraints.

For instance, with backward Euler one would get

$$\begin{aligned} \mathbf{y}_{n+1} &= \mathbf{y}_n + \Delta t\, \mathbf{F}(\mathbf{y}_{n+1},\mathbf{z}_{n+1}) \\ \mathbf{z}_{n+1} &= \mathbf{z}_n + \frac{1}{\epsilon}\Delta t\, \mathbf{G}(\mathbf{y}_{n+1},\mathbf{z}_{n+1}) \end{aligned}$$

The limit $\epsilon \to 0$ corresponds for the second equation to $\Delta t \to \infty$.

For backward Euler this should not lead to an instability for the backward Euler. This gives then

$$\begin{aligned} \mathbf{y}_{n+1} &= \mathbf{y}_n + \Delta t\, \mathbf{F}(\mathbf{y}_{n+1},\mathbf{z}_{n+1}) \\ 0 &= \mathbf{G}(\mathbf{y}_{n+1},\mathbf{z}_{n+1}) \end{aligned}$$

which is to be solved using Newton iterations. The function for which we need the root is then given by

$$\tilde{\mathbf{G}}(\mathbf{u},\mathbf{v}) = \begin{pmatrix} -\mathbf{u} + \mathbf{y}_n + \Delta t\, \mathbf{F}(\mathbf{u},\mathbf{v}) \\ \mathbf{G}(\mathbf{u},\mathbf{v}) \end{pmatrix}.$$

**Note:**

- We saw that even unconditionally stable schemes can be unsuitable for problems with very large time steps if they do not damp modes sufficiently strongly. For instance, Crank-Nicholson has vanishing damping in the limit $\Delta t \to \infty$.
- Like backward Euler, backward differencing schemes have strong damping for $\Delta t \to \infty$ and should be suitable and would provide higher order accuracy. E.g. BD2

  $$\begin{aligned} \frac{3}{2}\mathbf{y}_{n+1} - 2\mathbf{y}_n + \tfrac{1}{2}\mathbf{y}_n &= \mathbf{y}_n + \Delta t\, \mathbf{F}(\mathbf{y}_{n+1},\mathbf{z}_{n+1}) \\ \frac{3}{2}\mathbf{z}_{n+1} - 2\mathbf{z}_n + \tfrac{1}{2}\mathbf{z}_n &= \mathbf{z}_n + \frac{1}{\epsilon}\Delta t\, \mathbf{G}(\mathbf{y}_{n+1},\mathbf{z}_{n+1}) \end{aligned}$$

  Since they differ from backward Euler only in the time-derivative part they lead in the limit $\epsilon \to 0$ to the same equation for dealing with the algebraic constraint $\mathbf{G}(\mathbf{y}_{n+1},\mathbf{z}_{n+1})$, but to different equations for the differential equation.

**Implicit Runge-Kutta Method**[20]

[20] For history and a review see [2].

So far we only considered explicit Runge-Kutta methods. For DAE certain implicit Runge-Kutta methods provide higher-order stable schemes.

Recall the general form of the Runge-Kutta scheme

$$y_{n+1} = y_n + \Delta t \sum_{k=0}^{s} \gamma_k F_k$$

with

$$F_k = F\left(t_n + \alpha_k \Delta t, y_n + \Delta t \sum_{m=0}^{s} \beta_{km} F_m\right) \qquad k = 0, ..., s$$

which can also be written as

$$y_{n,k} = y_n + \Delta t \sum_{m=0}^{s} \beta_{km} F(t_n + \alpha_m \Delta t, y_{n,m}) \qquad k = 0 \ldots s$$

with

$$y_{n+1} = y_n + \Delta t \sum_{k=0}^{s} \gamma_k F\left(t_n + \alpha_k \Delta t, y_{n,k}\right)$$

The coefficients are given by the Butcher array

| | | | | | |
|---|---|---|---|---|---|
| $\alpha_0$ | $\beta_{00}$ | $\beta_{01}$ | $\ldots$ | $\ldots$ | $\beta_{0,s}$ |
| $\alpha_1$ | $\beta_{10}$ | $\beta_{11}$ | $\beta_{1,2}$ | | $\beta_{1,s}$ |
| $\alpha_2$ | $\beta_{20}$ | $\beta_{21}$ | $\beta_{22}$ | | $\ldots$ |
| ... | ... | ... | ... | | $\ldots$ |
| $\alpha_s$ | $\beta_{s,0}$ | ... | ... | $\beta_{s,s-1}$ | $\beta_{s,s}$ |
| | $\gamma_0$ | $\gamma_1$ | ... | $\gamma_{s-1}$ | $\gamma_s$ |

For the explicit methods we had $\alpha_0 = 0$ and $\beta_{ij} = 0$ for $j \geq i$. Then one could first determine $F_0$, then $F_1$ etc. and then combine them to update $y_{n+1}$.

In the implicit case all contributions $F_k$ have to be computed simultaneously rather than sequentially, again by Newton if the equations are nonlinear.

For the index-1 DAE, considering them as arising in the limit of $\epsilon \to 0$, one has to satisfy two types of equations,

$$\begin{aligned} \mathbf{y}_{n,k} &= \mathbf{y}_n + \Delta t \sum_{m=0}^{s} \beta_{km} \mathbf{F}(t_n + \alpha_m \Delta t, \mathbf{y}_{n,m}, \mathbf{z}_{n,m}) \qquad k = 0 \ldots s \\ 0 &= \sum_{m=0}^{s} \beta_{km} \mathbf{G}(t_n + \alpha_m \Delta t, \mathbf{y}_{n,m}, \mathbf{z}_{n,m}) \qquad k = 0 \ldots s \end{aligned}$$

the solution of which is then combined to

$$\mathbf{y}_{n+1} = \mathbf{y}_n + \Delta t \sum_{k=0}^{s} \gamma_k \mathbf{F}\left(t_n + \alpha_k \Delta t, \mathbf{y}_{n,k}, \mathbf{z}_{n,k}\right).$$

How to update $\mathbf{z}_n$? In the limit of $\Delta t \to \infty$ the analogous update cannot be done for $\mathbf{z}_n$.

If the matrix $\beta_{km}$ is invertible, the algebraic equations imply that the constraint is satisfied by each stage on its own,

$$\mathbf{G}(t_n + \alpha_m \Delta t, \mathbf{y}_{n,m}, \mathbf{z}_{n,m}) = 0 \qquad m = 0 \dots s$$

Therefore, if $\mathbf{y}_{n,s} = \mathbf{y}_{n+1}$, the update

$$\mathbf{z}_{n+1} = \mathbf{z}_{n,s}$$

would satisfy the constraint at time $t_{n+1}$.

To have $\mathbf{y}_{n,s} = \mathbf{y}_{n+1}$ one needs $\alpha_s = 1$, i.e. the last stage has to be at $t_{n+1}$, and the last row of the matrix $\beta_{km}$ has to be equal to the vector $\gamma_m$

$$\gamma_m = \beta_{sm} \qquad m = 0 \dots s$$

Such methods are called *stiffly accurate*. A popular, stiffly accurate method is the 5th-order Radau[21] 5 method defined by the Butcher array

$$\begin{array}{c|ccc} \frac{4-\sqrt{6}}{10} & \frac{88-7\sqrt{6}}{360} & \frac{296-169\sqrt{6}}{1800} & \frac{-2+3\sqrt{6}}{225} \\ \frac{4+\sqrt{6}}{10} & \frac{296+169\sqrt{6}}{1800} & \frac{88+7\sqrt{6}}{360} & \frac{-2-3\sqrt{6}}{225} \\ 1 & \frac{16-\sqrt{6}}{26} & \frac{16+\sqrt{6}}{26} & \frac{1}{9} \\ \hline & \frac{16-\sqrt{6}}{26} & \frac{16+\sqrt{6}}{26} & \frac{1}{9} \end{array}$$

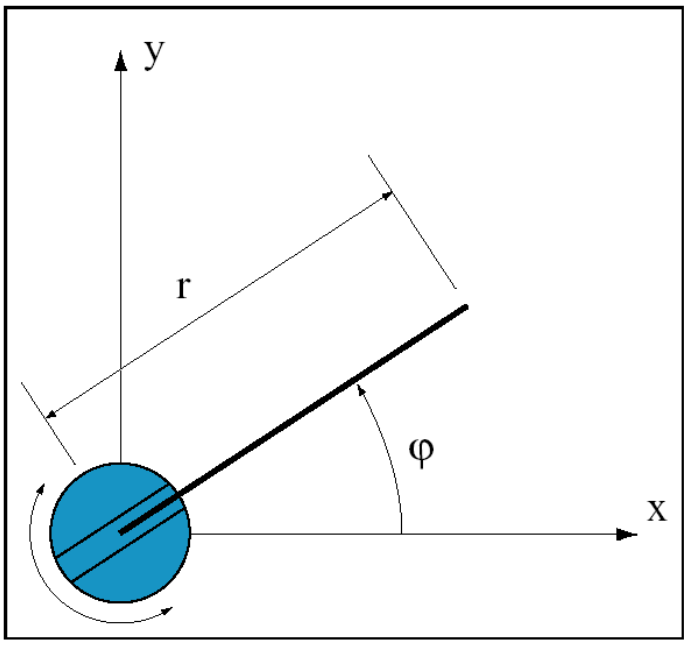
**Note:**

- In Radau methods the grid points ($\alpha_m$) are optimized with the condition that either the right end point or the left end point is fixed. For the Radau method above $\alpha_s = 1$ is imposed.

[21] Rodolphe Radau, 1880

# 15 Boundary Value Problems Application 2: Control of Robot

Consider a model for a simple robot consisting of an extendable arm on a pivot




It is described by

- extension $r$ of the arm
- angle $\phi$ of arm relative to $x-$axis
- radial velocity $\frac{dr}{dt}$
- angular velocity $\frac{d\phi}{dt}$

Newton's law of motion

$$\begin{aligned} \frac{d}{dt}\left((I+mr^2)\frac{d\phi}{dt}\right) &= \tau(t) \\ \frac{d}{dt}\left(m\frac{dr}{dt}\right) &= f(t)+mr\left(\frac{d\phi}{dt}\right)^2 \end{aligned}$$

- $\tau$ external applied torque
- $f$ applied radial force
- centrifugal force $mr\left(\frac{d\phi}{dt}\right)^2$

- $I$ is the moment of inertia of the arm with respect to an axis through its center of mass
- $m$ the mass of the arm.

For numerical treatment write in terms of first-order equations

$$\frac{d\phi}{dt} = \omega \tag{68}$$
$$\frac{dr}{dt} = v \tag{69}$$
$$\frac{d\omega}{dt} = \frac{1}{I_{eff}(r)}\left(\tau(t) - 2mr\omega v\right) \tag{70}$$
$$\frac{dv}{dt} = \frac{f(t)}{m} + r\omega^2 \tag{71}$$

with

$$I_{eff}(r) = I + mr^2$$

The goal is to determine the time-dependent force and torque such that the arm moves from an initial position

$$[r(0), \phi(0), \frac{dr}{dt}(0), \frac{d\phi}{dt}(0)] = [r_0, \phi_0, v_0, \omega_0]$$

to a final position

$$[r(t_f), \phi(t_f), \frac{dr}{dt}(t_f), \frac{d\phi}{dt}(t_f)] = [r_f, \phi_f, v_f, \omega_f]$$

at a time $t_f$.
There are inifinitely many ways to do that.
To make the problem meaningful introduce a *cost function*

$$P = \int_0^{t_f} E(\tau, F, ...)\, dt$$

Now: find solution that goes from initial position to final position and *minimizes* the cost function.

Here use the cost function

$$P = \frac{1}{2}\int_0^{t_f} \alpha\, \tau(t)^2 + \frac{1}{\alpha}\, f(t)^2\, dt$$

**Note:**

- Why not take the work performed by the robot?
  Accelerating and breaking usually both require effort although during breaking energy is extracted from robot $\Rightarrow$ take quadratic form in the forces.

- With the parameter $\alpha$ we can emphasize more whether applying a torque or a linear force requires more resources.

Apply now the results from the single dynamical variable to our robot equations.

- The dynamical system is given by four variables $(\phi(t), r(t), \omega(t), v(t))$. Thus there are 4 variables $x^{(i)}(t)$ which satisfy the 4 constraints given by the original dynamical system (68,69,70,71),

$$\frac{dx^{(i)}}{dt} = F^{(i)}(x^{(j)}, u^{(j)}) \qquad i = 1 \dots 4.$$

To satisfy these four constraints we need four Lagrange parameters $\lambda^{(i)}(t)$. Using discrete time steps as in (46), eq.(45) becomes then

$$\nabla \left( \sum_{n=1}^{N} E\left(\tau_n, f_n\right) + \sum_{k=1}^{4} \sum_{n=1}^{N} \lambda_n^{(k)} g_n^{(k)} \right) = 0.$$

Thus, the 4 constraints $g^{(k)}$ defining the dynamical system are added together. Here we have made use of the fact that in our robot problem $P$ does not depend on $x$.
Here

$$\nabla = \left( \frac{\partial}{\partial x_1^{(1)}}, \dots, \frac{\partial}{\partial x_n^{(k)}}, \dots \frac{\partial}{\partial x_N^{(4)}}, \frac{\partial}{\partial \tau_1}, \dots, \frac{\partial}{\partial f_N} \right).$$

- Comparison with (46) and (48) shows that in continuous time the Lagrange parameters $\lambda^{(i)}(t)$ satisfy the evolution equations

$$\frac{d\lambda^{(i)}}{dt} = -\left( \sum_{k=1}^{4} \lambda^{(k)}(t) \frac{\partial F^{(k)}(x^{(1)}(t), x^{(2)}(t), x^{(3)}(t), x^{(4)}(t), \tau(t), f(t))}{\partial x^{(i)}} + \underbrace{\frac{\partial E(\tau(t), f(t))}{\partial x^{(i)}}}_{=0} \right), \qquad i = 1 \dots \tag{72}$$

- There are two control variables

$$u^{(1)}(t) = \tau(t) \qquad u^{(2)}(t) = f(t)$$

They are related to the Lagrange parameters via (50)

$$\sum_{k=1}^{4} \lambda^{(k)} \frac{\partial F_k(x^{(j)}, \tau, f)}{\partial \tau} + \frac{\partial E(\tau, f)}{\partial \tau} = 0 \tag{73}$$

$$\sum_{k=1}^{4} \lambda^{(k)} \frac{\partial F_k(x^{(j)}, \tau, f)}{\partial f} + \frac{\partial E(\tau, f)}{\partial f} = 0 \tag{74}$$

Evaluating (72,73,74) we obtain

$$\frac{d\lambda^{(1)}}{dt} = 0 \tag{75}$$

$$\frac{d\lambda^{(2)}}{dt} = \frac{2m}{I_{eff}(r)}\left(\omega v + r\frac{d\omega}{dt}\right)\lambda^{(3)} - \omega^2\lambda^{(4)} \tag{76}$$

$$\frac{d\lambda^{(3)}}{dt} = -\lambda_1 + \frac{2m}{I_{eff}(r)}\, rv\lambda^{(3)} - 2r\omega\lambda^{(4)} \tag{77}$$

$$\frac{d\lambda^{(4)}}{dt} = -\lambda^{(2)} + \frac{2m}{I_{eff}(r)}\, r\omega\lambda^{(3)} \tag{78}$$

with

$$\tau = \frac{1}{\alpha}\frac{\lambda^{(3)}}{I_{eff}} \qquad F = \alpha\frac{\lambda^{(4)}}{m} \tag{79}$$

**Now:**

- Known are the initial and final conditions for $\phi$, $r$, $\omega$, $v$
- Unknown are the initial conditions for $\lambda^{(i)}$, $i = 1...4$
- $\Rightarrow$ total count is o.k.: 8 equations for 8 variables

Now the situation is similar to that of positioning a satellite:
Some of the initial conditions have been replaced by final conditions.

# 16 Applications

## 16.1 Application 1a: Fluid Flow: Vortex Dynamics

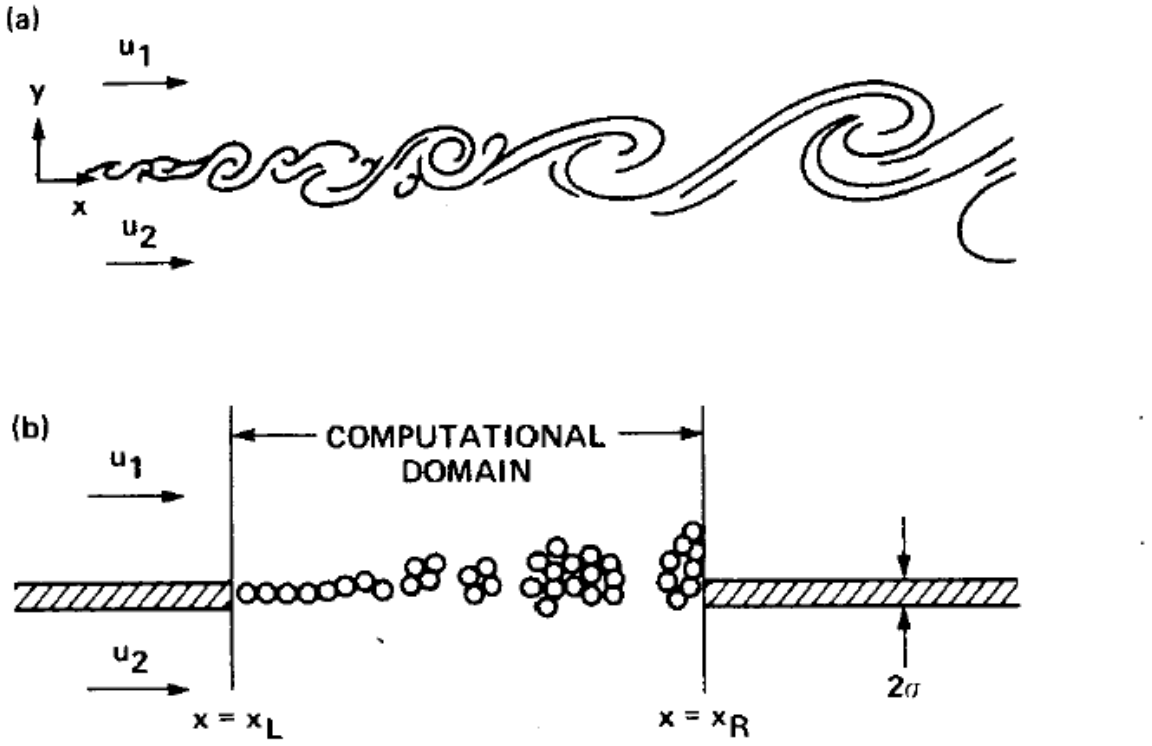
It turns out that for simulations of smooth vorticity fields point vortices are not suited. Approximating sheets of vorticity in a mixing layer by lines of vortices leads to singularities.

Improvement is obtained by using blob vortices, i.e. vortices with finite extent.

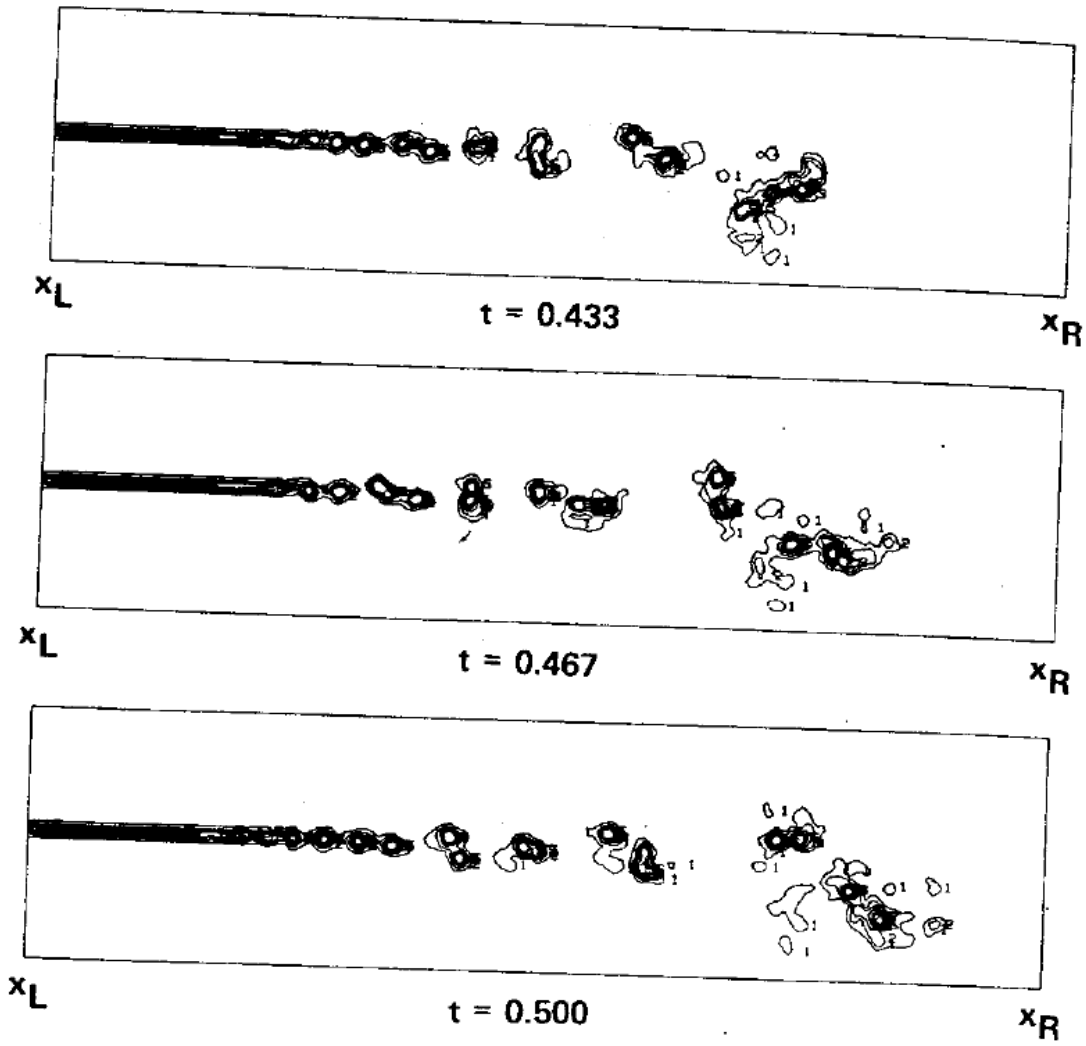
Example simulation from Leonard (J. Comp. Phys. 37 (1980) 289).

Schematic set-up of mixing layer: inflow of two layers of fluid with two different velocities, mixing occurs starting at the end of separating plate

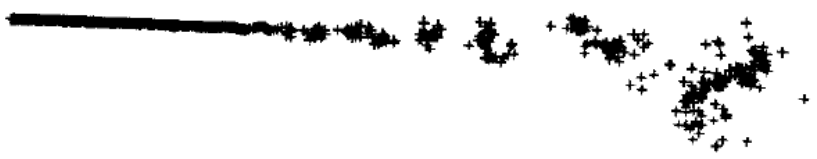
Vortex blobs are created numerically at the end of the separation plate




Vorticity contours of simulation

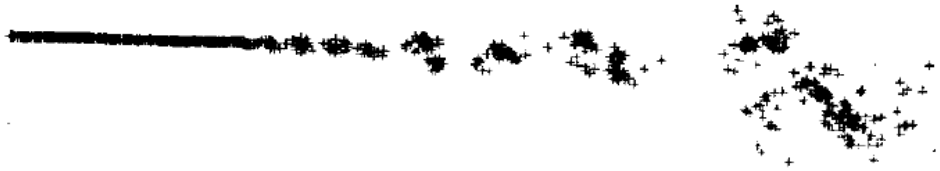



Corresponding positions of vortex blobs



t = 0.433

t = 0.467



t = 0.500

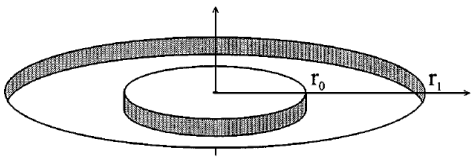
## 16.2 Application 2: Ostwald Ripening and the Decay of Islands on Surfaces

Consider the surface of a crystaline material during deposition of material from the vapor phase:

- Typically the surface is flat (terraces) except for mono-atomic steps between the terraces. The terraces form islands.
- At sufficiently high (but not too high) temperatures individual atoms on the surface (adatoms=adsorbed atoms) are diffusing along the surface. (At yet higher temperatures the terraces cease to be flat). The adatoms tend to preferentially attach at the step edges (more bonds established than on flat surface)
  adatom can more easily attach to concavely curved surface than to convex surface: 'larger contact area'

  $\Rightarrow$it is observed that small islands shrink and large islands grow reflecting the smaller convex curvature of the large islands.
- In principle, the adatoms can also jump up or down the steps between terraces. In many materials these jumps are relatively rare.

Consider simple model for the decay of a single island in a pit




- Diffusion is fast enough compared to growth/decay of islands to be in steady state
$$\partial_t \rho = \Delta \rho \qquad \Rightarrow \qquad \Delta \rho = 0. \tag{80}$$
  $\rho$ is the density of atoms diffusing on the surface (terrace) between steps.
- Assume dynamics on each terrace independent of $\rho$ on adjacent islands: *neglect jumps between terraces*; to hop down or even up a step the atoms would have to overcome the Ehrlich barrier.
- Atoms can attach to step but also detach from the step: the more atoms diffusing around the more likely they are to attach to the step
  $\rightarrow$ for some density of diffusing atoms as many atoms attach as detach: equilibrium

$\rightarrow$ net attachment rate of adatoms to up-step proportional to deviation of $\rho$ from equilibrium value

$$K\left(\rho - \rho^{eq}(R)\right)$$

atoms attach if $\rho > \rho^{eq}$, i.e. if density of 'gas' too high, $K > 0$.
If the atoms do not like to attach to the step a higher density is needed keep the atoms from detaching $\rightarrow$equilibrium value $\rho^{eq}(R)$ is higher for more strongly curved steps, i.e. for smaller radius of curvature $R$ of the step

$$\rho^{eq}(R) = \rho_{\infty}^{eq}\, e^{\frac{\gamma\Omega}{k_B T}\frac{1}{R}}$$

$\gamma$ line tension (= one-dimensional surface tension), $\Omega$ surface area covered by single atom, $k_B$ Boltzmann constant, $T$ temperature. $\rho_{\infty}^{eq}$ is the equilibrium density for a straight step.
**Note:**

- $R < 0$ concave step: atoms more attracted to wall than for straight wall, even for smaller density atoms attach easily.
- $R > 0$ convex step: atoms less likely to attach, higher density needed to have them attach.

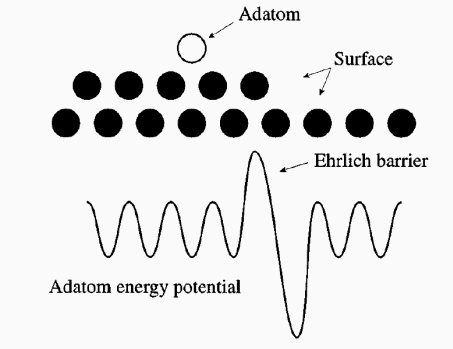
- attaching adatoms supplied by diffusive flux $j_D$ from dense to dilute regions

$$j_D = -D\frac{d\rho}{dx}$$

with $D$ diffusion constant. Here for radial geometry

$$|j_D| = \left|-D\frac{d\rho}{dn}\right| = \left|K\left(\rho - \rho^{eq}(R)\right)\right|$$

with $\frac{d}{dn}$ normal derivative
We will determine signs later depending on which orientation the step has.

- 


To diffuse, the atoms need to detach from the surface, hop over, and reattach at a neighboring location.
$\rightarrow$ temperature dependence of diffusion

$$D = D_0 e^{-\frac{E_d}{kT}}$$

with $E_d$ surface diffusion barrier (activation energy for detaching/breaking bond with atom on surface)

- When attaching the atoms gain a binding energy: $E_f$ adatom formation energy
  $\rightarrow$ Temperature dependence of equilibrium density

$$\rho_\infty^{eq} = C_0 e^{-\frac{E_f}{kT}}$$

  for higher $E_f$ the atoms attach more readily and fewer atoms are in the gas phase.

- attachment of adatoms moves the edge

$$\begin{aligned} \frac{dr_0(t)}{dt} &= \kappa D \frac{d\rho(r_0)}{dr} \\ \frac{dr_1(t)}{dt} &= \kappa D \frac{d\rho(r_1)}{dr} \end{aligned}$$

  At $r_0$ island shrinks for $\frac{d\rho}{dr} < 0$; at $r_1$ pit shrink for $\frac{d\rho}{dr} < 0$.

To get an evolution equation for the radii $r_{0,1}(t)$ we need the densities $\rho(r_0)$ and $\rho(r_1)$: determine the density profile from Laplace equation (80).
For simplicity: assume circular island in circular pit

$$\Delta\rho \equiv \frac{d^2\rho}{dr^2} + \frac{1}{r}\frac{d\rho}{dr} = 0$$

Need boundary conditions

$$\begin{aligned} -D\frac{d\rho}{dr}(r_0) &= \text{flux out of the inner step} \\ -D\frac{d\rho}{dr}(r_1) &= \text{flux into the outer step} \end{aligned}$$

Signs of fluxes: for $\rho > \rho^{eq}$ adatoms *attach* to the step

- at $r_0$: negative flux

$$-D\frac{d\rho}{dr}(r_0) = -K\left(\rho(r_0) - \rho^{eq}(r_0)\right)$$

- at $r_1$: positive flux

$$-D\frac{d\rho}{dr}(r_1) = +K\left(\rho(r_1) - \rho^{eq}(r_1)\right)$$

Seek evolution equations for $r_{0,1}(t)$:

General solution for diffusion equation:

$$\rho = A + B \ln r$$

(Obtain via $v = \frac{d\rho}{dr}$: $v' = -v/r \Rightarrow dv/v = -dr/r \Rightarrow v \sim 1/r \Rightarrow \rho \sim \ln r$)

Insert

$$\begin{aligned} D\frac{B}{r_0} &= -K\left(\left(\rho^{eq}(r_0) - A - B\ln r_0\right)\right) \\ D\frac{B}{r_1} &= K\left(\left(\rho^{eq}(r_1) - A - B\ln r_1\right)\right) \end{aligned}$$

need only $\frac{d\rho}{dr} \propto B$:

$$\begin{aligned} -A &= \frac{DB}{Kr_1} - \rho^{eq}(r_1) + B\ln r_1 \\ -\frac{DB}{r_0 K} &= \rho^{eq}(r_0) + \frac{DB}{Kr_1} - \rho^{eq}(r_1) + B\ln r_1 - B\ln r_0 \end{aligned}$$

$$\begin{aligned} B(r_0, r_1) &= \frac{\rho^{eq}(r_1) - \rho^{eq}(r_0)}{\frac{D}{K}(\frac{1}{r_0} + \frac{1}{r_1}) + \ln\frac{r_1}{r_0}} = \frac{Kr_0 r_1\left(\rho^{eq}(r_1) - \rho^{eq}(r_0)\right)}{D(r_0 + r_1) + Kr_0 r_1 \ln\frac{r_1}{r_0}} < 0 \\ \rho^{eq}_{r_0} &= \rho^{eq}_{\infty}\, e^{\frac{\gamma\Omega}{kT}\frac{1}{r_0}} \\ \rho^{eq}_{r_1} &= \rho^{eq}_{\infty}\, e^{-\frac{\gamma\Omega}{kT}\frac{1}{r_1}} \end{aligned}$$

Insert general solution in evolution equations for radii

$$\frac{dr_0(t)}{dt} = \kappa DB(r_0, r_1)\,\frac{1}{r_0} \qquad \frac{dr_1(t)}{dt} = \kappa DB(r_0, r_1)\,\frac{1}{r_1}$$

**Notes:**

- as islands shrinks $r_0 \to 0$ then $B \to -\infty$ exponentially fast through $\rho^{eq}(r_0)$: evolution becomes very fast in the final phase
- adaptive time step should be useful.

# 17 Neuronal Action Potentials - Hodgkin-Huxley Model

The brain has $\mathcal{O}(10^{11})$ neurons. What do they do?

A few, very basic aspects of their function:

- Most neurons communicate with each other by firing electric action potentials.
  The neurons can be *excitable* and fire only when their input reaches a threshold or they fire spontaneously and their firing rate is modulated by the strength of their inputs.
- Most neurons receive *input* from thousands of other neurons: that input can be excitatory *and* inhibitory for a given neuron
- Most neurons send their *output* to thousands of other neurons: the output from a given neuron is *either* excitatory *or* inhibitory (Dale's principle), i.e. there are excitatory neurons and inhibitory neurons.
- Due to the firing threshold the neural response to excitatory inputs yields something like an AND function;
  the response to inhibitory inputs is something like an AND NOT function
- Depending on their response speed neurons can respond to quite different signals:
  - Slow neurons accumulate their input over an extended period of time and their activity extracts the long-time behavior of their input. These neurons function as 'integrators'.
  - Fast neurons do not accumulate their input over long time and therefore require the inputs to occur almost simultaneously to evoke a spike. These neurons perform 'coincidence detection' ; they can select very specific inputs from a large collection of inputs.

Neurons have spatial structure:

- they receive their input on their dendrites, which typically are quite ramified.
- they send their output through axons to other neurons, which can be very far away ('from head to toe').
- the voltage in a neuron depends on space

Here we focus on a small piece of membrane of a neuron and model action potentials and excitability. We discuss the model developed by Hodgkin and Huxley (1952), which has laid the foundation to the mathematical description of the biophysical properties of neurons and how they process information in the brain.

**Membrane:**

- The electrical state of a neuron is characterized foremost by the electrical voltage across its membrane
- The membrane separates regions with very different ion concentrations
  - outside the cell the $Na^+$-concentration is high and the $K^+$-concentration is low
  - inside the cell the opposite is the case: $[Na^+]$ is low and $[K^+]$ is high.
  - the cell is kept in this non-equilibrium state by slow pumps
- The membrane is active: it contains ion channels that are selectively permeable only for certain types of ions. Particularly relevant are channels for $Na^+$ and $K^+$. These ion channels can open and close depending on the voltage across the membrane. Thus, depending on the voltage different ion currents flow across the membrane.
- The membrane is a capacitor:
  - currents across the membrane $\Rightarrow$ change in the charge on the capacitor
  - change in the charge $\Rightarrow$ change in the voltage across the membrane
  - the total current across the membrane consists of various ionic currents Kirchhoff's law yields then the differential equation for the voltage
- –

$$C\,\frac{dV}{dt} = -I_{Na^+} - I_{K^+} - I_{leak} - I_{inj} \tag{81}$$

**Notation:** positive currents are outward: from inside the cell to outside the cell
$\Rightarrow$ positive currents make the voltage inside more negative: they hyperpolarize the cell.

Potassium current $I_{K^+}$

$$I_{K^+} = \bar{G}_K\, n(t)^4\, (V(t) - E_K)$$

- To be open the channel requires 4 'particles' (~molecules) to be in a specific configuration; the probability for each of the particles to be in that configuration is $n$. Averaging over many channels the fraction of channels that are open is given by $n$.

The operation of the channels can be described using transition rates $\alpha_n$ and $\beta_n$ between the two configurations, which depend on the voltage

$$\frac{dn}{dt} = \alpha_n(V)\,(1-n) - \beta_n(V)\,n \tag{82}$$

with

$$\alpha_n(V) = \frac{V+55}{100\;(1-e^{-(V+55)/10})} \qquad \beta_n(V) = 0.125\,e^{-0.0125(V+65)}$$

where $V$ is measured in $mV$.
We can rewrite the equation for $n$ as

$$\tau_n \frac{dn}{dt} = n_\infty - n$$

with (cf. Fig.24b)

$$\tau_n(V) = \frac{1}{\alpha_n + \beta_n} \qquad n_\infty(V) = \frac{\alpha_n}{\alpha_n + \beta_n}$$

**Note:** The functional form of $\alpha_n(V)$ and $\beta_n(V)$ as well as the numerical values were obtained by fitting to experiments. Different neurons have somewhat different parameter values.

- Because the $K^+$-concentration is different inside and outside the cell a $K^+$-current flows even for vanishing voltage difference. In other words, for the current to vanish one has to apply a sufficiently large voltage across the membrane,

  $$E_K = -77mV.$$

  $E_K$ is called the *reversal potential* of this channel. Even for slightly negative voltage a $K^+-$current flows outward.

- This $K^+$-current is persistent: it remains on as long as the cell is depolarized (voltage positive).

Sodium current $I_{Na^+}$

$$I_{Na^+} = \bar{G}_{Na} m^3(t) h(t)\;(V(t) - E_{Na}) \qquad E_{Na} = +50mV$$

The sodium current is controled by two different 'particles'

- the activation particle $m$ opens with increasing voltage. This activation is *fast*.

$$\frac{dm}{dt} = \alpha_m(V)\,(1-m) - \beta_m(V)\,m \tag{83}$$

$$\alpha_m(V) = \frac{V+40}{10\;(1-e^{-(V+40)/10})} \qquad \beta_m(V) = 4\,e^{-0.0556(V+65)}$$

- the inactivation particle $h$ closes with increasing voltage. The inactivation is relatively *slow*

$$\frac{dh}{dt} = \alpha_h(V)\,(1-h) - \beta_h(V)\,h \tag{84}$$

$$\alpha_h(V) = 0.07\,e^{-0.05(V+54)} \qquad \beta_h(V) = \frac{1}{1+e^{-0.1(V+35)}}.$$

Because of the inactivation variable $h$ the sodium current is *transient*: it opens when the membrane is depolarized, but even if the cell were to remain depolarized it would turn off after a while due to inactivation ($h$ goes to 0).

The action potential is driven by the following mechanism:

1. at rest the potential $V$ is strongly negative: $V_{rest} \sim -65mV$ .
2. when an input increases the voltage *sufficiently far above* $V_{rest}$ (towards positive voltage) the activation variable $m$ of $I_{Na^+}$ turns on rapidly (cf. time scales of the activation and inactivation variables in Fig.24b): $Na^+$ ions flow into the cell ($I_{Na^+} < 0$) and raise the voltage further providing positive feed-back and amplification of the input.
3. at this depolarized voltage the inactivation variable $h$ starts to decrease, but it does so much more slowly: $I_{Na^+}$ is *slowly* turned off. Note, that $I_{Na^+}$ decreases also because its driving force $V - E_{Na}$ decreases as the voltage increases.
4. the activation variable $n$ of $I_{K^+}$ also becomes activated: $K^+$ ions flow out of cell bringing the voltage down and, in fact, leading to an overshoot
5. $Na^+$ and $K^+$ ions are pumped back to where they came from and $V$ goes back to $V_{rest}$.

If the input is not strong enough to activate the $Na^+$-current the voltage changes only slightly and no action potential is fired.

**Notes:**

- After an action potential the cell cannot be activated again for some time since the inactivation variable $h$ has not recovered yet; the $Na^+$channels remain inactivated (closed): the cell has a *refractory period*.
- For steady input current the cell fires periodically with a frequency that increases with the input current.

# References

[1] Toni Feder. Statistical physics for the birds. *Physics Today*, October:28, 2007.

[2] E. Hairer and G. Wanner. Stiff differential equations solved by Radau methods. *Journal of Computational and Applied Mathematics*, 111(1-2):93–111, November 1999.

[3] H. Levine, W. J. Rappel, and I. Cohen. Self-organization in systems of self-propelled particles. *Phys. Rev. E*, 6302(2):017101, January 2001.

# 18 Connection between Stability and Convergence

**Definition:**

A one-step scheme

$$y_{n+1} = y_n + \Delta t \mathcal{F}(t_n, y_n, \Delta t)$$

is called *consistent* for the differential equation $dy/dt = F(t, y)$ if

$$\lim_{\Delta t \to 0} \mathcal{F}(t_n, y_n, \Delta t) = F(t_n, y_n)$$

**Theorem:**

A scheme that is *consistent* and *stable* converges to the exact solution in the limit $\Delta t \to 0$.

**Note:**

- in each time step truncation and round-off errors arise. We need that they do not accumulate catastrophically, but remain bounded.

**Notes:**

- For linear differential equations

  $$\frac{d\mathbf{y}}{dt} = \mathbf{L}\mathbf{y} + \mathbf{b}$$

  it is sufficient to consider only homogeneous equation. Inhomogeneous equation has general solution

  $$\mathbf{y} = \mathbf{y}_h(t) + \mathbf{y}_p(t)$$

  where

  $$\frac{d\mathbf{y}_p}{dt} = \mathbf{L}\mathbf{y}_p + \mathbf{b} \qquad \frac{d\mathbf{y}_h}{dt} = \mathbf{L}\mathbf{y}_h$$

  Consider implementing the equation for $\mathbf{w} = \mathbf{y} - \mathbf{y}_p(t)$ with $\mathbf{y}_p(t)$ given,

  $$\frac{d\mathbf{w} + \mathbf{y}_p}{dt} = \mathbf{L}(\mathbf{w} + \mathbf{y}_p) + \mathbf{b} \qquad \Rightarrow \frac{d\mathbf{w}}{dt} = \mathbf{L}\mathbf{w}$$

**Proof only for one-step schemes for homogeneous equation:**

Go through proof to see how stability and consistency enter the convergence

For linear homogeneous equations the solution to the numerical scheme can be written formally

$$y(t + \Delta t) = S(t + \Delta t, t)\, y(t)$$

e.g. forward Euler for $\frac{dy}{dt} = \lambda(t)y$

$$S(t + \Delta t, t) = 1 + \Delta t\, \lambda(t)$$

analogously for coupled equations and Runge-Kutta etc.

For stable scheme we have

$||y(t+\Delta t)|| \leq ||y(t)|| \qquad \Rightarrow ||S(t+\Delta t, t)y(t)|| \leq ||y(t)||$ for all $y(t)$ i.e. $||S(t+\Delta t, t)|| \leq 1$

The exact solution $\hat{y}(t)$ satisfies

$$\hat{y}(t + \Delta t) = S(t + \Delta t, t)\hat{y}(t) + \tau(t + \Delta t) + \epsilon$$

with the truncation error $\tau(t) = \mathcal{O}(\Delta t^p) = C(t)\, \Delta t^p$ and a round-off error $\epsilon$.

Need to compute error at final time $t_{max} = N\Delta t$

Consider first

$$\begin{aligned} y(2\Delta t) &= S(2\Delta t, \Delta t)\, (y(\Delta t)) = \\ &= S(2\Delta t, \Delta t)\, S(\Delta t, 0)\, y(0) \end{aligned}$$

Thus

$$y(N\Delta t) = \prod_{j=1}^{N} S(j\Delta t, (j-1)\Delta t)\, y(0)$$

For simplicity assume: differential equation does not depend explicitly on time

$$S(t + \Delta t, t) = S(\Delta t) \qquad \text{independent of } t$$

$$S(N\Delta t) = (S(\Delta t, 0))^N \equiv S^N$$

Consider error $e(t) = \hat{y}(t) - y(t)$:

$$\begin{aligned} e(t + \Delta t) &= \hat{y}(t + \Delta t) - y(t + \Delta t) = S\, \hat{y}(t) + \tau(t + \Delta t) + \epsilon - S\, y(t) = \qquad \text{since linear} \\ &= S\, e(t) + \tau(t + \Delta t) + \epsilon \end{aligned}$$

Thus:

$$\begin{aligned} e(\Delta t) &= S\, e(0) + \tau(\Delta t) + \epsilon_1 \\ e(2\Delta t) &= S\, e(\Delta t) + \tau(2\Delta t) + \epsilon_2 = S^2 e(0) + S\, \{\tau(\Delta t) + \epsilon_1\} + \tau(2\Delta t) + \epsilon_2 \end{aligned}$$

Note that the round-off error is expected to be different in each time step, $\epsilon_j \neq \epsilon_i$ for $j \neq i$.

Then, iterating this procedure one obtains

$$\begin{aligned} e(N\Delta t) &= S^N\, e(0) + S^{N-1}\left\{\tau(\Delta t) + \epsilon_1\right\} + S^{N-2}\left\{\tau(2\Delta t) + \epsilon_2\right\} + ... + \tau(N\Delta t) + \epsilon_N \\ &= S^N\, e(0) + \sum_{j=1}^{N} S^{N-j}\ \left\{\tau(j\Delta t) + \epsilon_j\right\} \end{aligned}$$

Estimate the size of error

$$\begin{aligned} ||e(N\Delta t)|| &\leq ||S^N\, e(0)|| + \sum_{j=1}^{N} ||S^{N-j}\ \left\{\tau(j\Delta t) + \epsilon_j\right\}|| \\ &\leq \|e(0)\| + \sum_{j=1}^{N} \|\tau(j\Delta t) + \epsilon_j\| \qquad \text{because of stability} \end{aligned}$$

The scheme is assumed to be consistent: $\tau = \mathcal{O}(\Delta t^p)$ with $p > 1$

$$||e(N\Delta t)|| \leq ||e(0)|| + \left[\mathcal{O}(\Delta t^{p-1}) + \frac{\max_j \{\epsilon_j\}}{\Delta t}\right] \underbrace{\sum_{j=1}^{N} \Delta t}_{t_{max}} = ||e(0)|| + t_{max}\left\{\mathcal{O}(\Delta t^{p-1}) + \frac{\max_j \{\epsilon_j\}}{\Delta t}\right\}$$

**Note:**

- Consistency ensures that truncation error in each time step is *smaller* than $\mathcal{O}(\Delta t)$ (e.g. $\mathcal{O}(\Delta t^2)$)
- Stability ensures that the local errors that arise in each time step are not amplified too much so that their accumulation goes to 0 for $\Delta t \to 0$ (e.g. $\mathcal{O}(\Delta t)$).
- This result can be shown more generally for nonlinear equations as well.

**Thus:** The scheme converges to the exact solution with the power of $\Delta t$ expected based on the truncation error.

Of course, the round-off error $\epsilon$ limits the accuracy. Worse yet, since it accumulates with each time step, it leads to a *reduction* in accuracy with *decreasing* time step if the time step is smaller than a minimal time step. For instance, applying the improved Euler scheme (RK2) (1) to the differential equation $y' = \lambda y$ one gets for the truncation error

$$\tau_j = \frac{1}{6}\Delta t^3 \lambda^3 y_j + \mathcal{O}(\Delta t^4)$$

and for double-precision the round-off error $\epsilon_j$, which is $\mathcal{O}(10^{-16}y)$, becomes of the same order as the truncation error for

$$\Delta t_{opt}^3 = \frac{6}{\lambda^3 y_j}\epsilon_j \qquad \Rightarrow \qquad \Delta t_{opt} = \mathcal{O}(10^{-5}) \tag{85}$$

**Note:**

- For implicit schemes for which a nonlinear equation has to be solved in each time step (cf. (16) below) the error incurred by solving this equation only approximately effectively contributes to the round-off error.

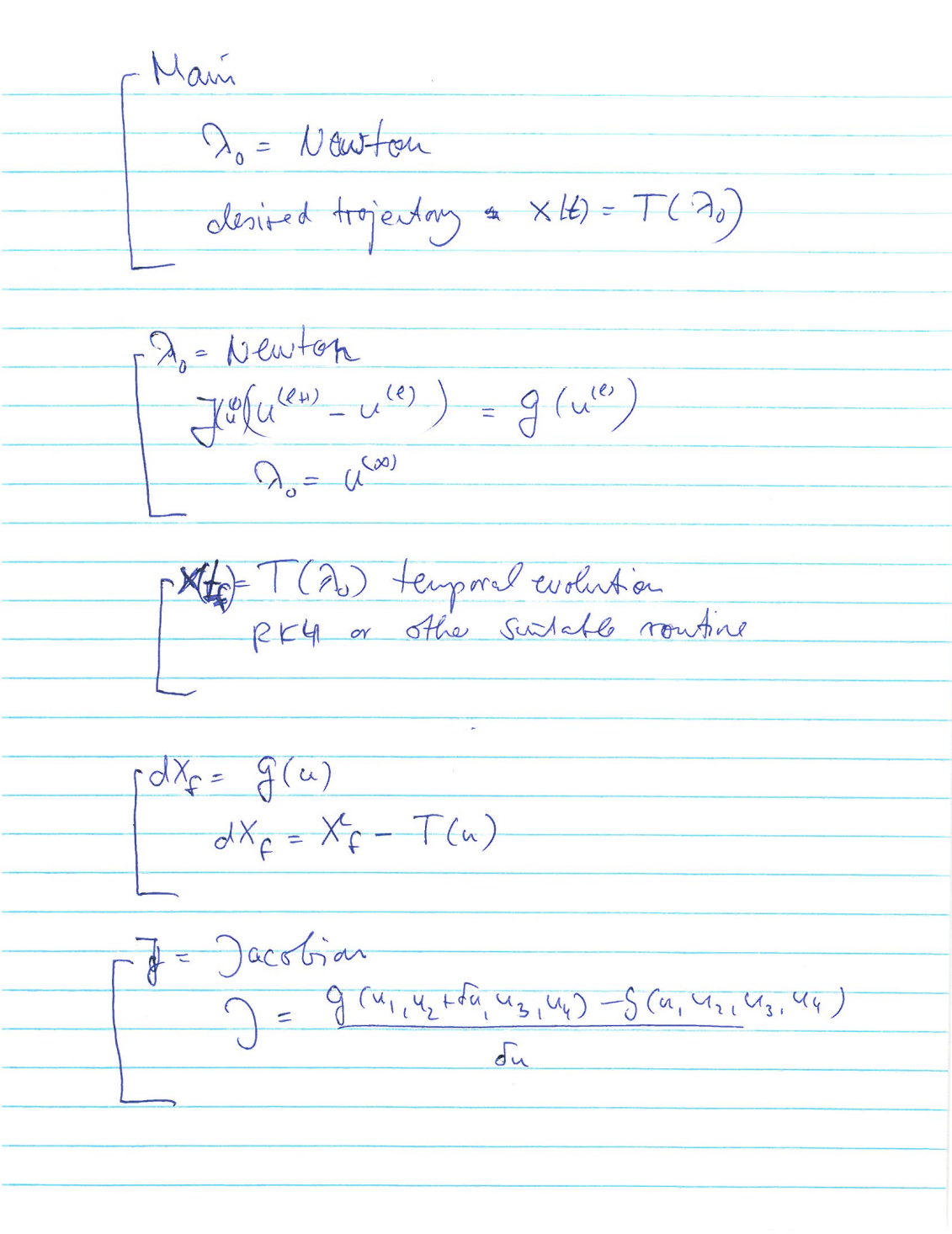



Figure 23: Pseudo-code for robot optimization.

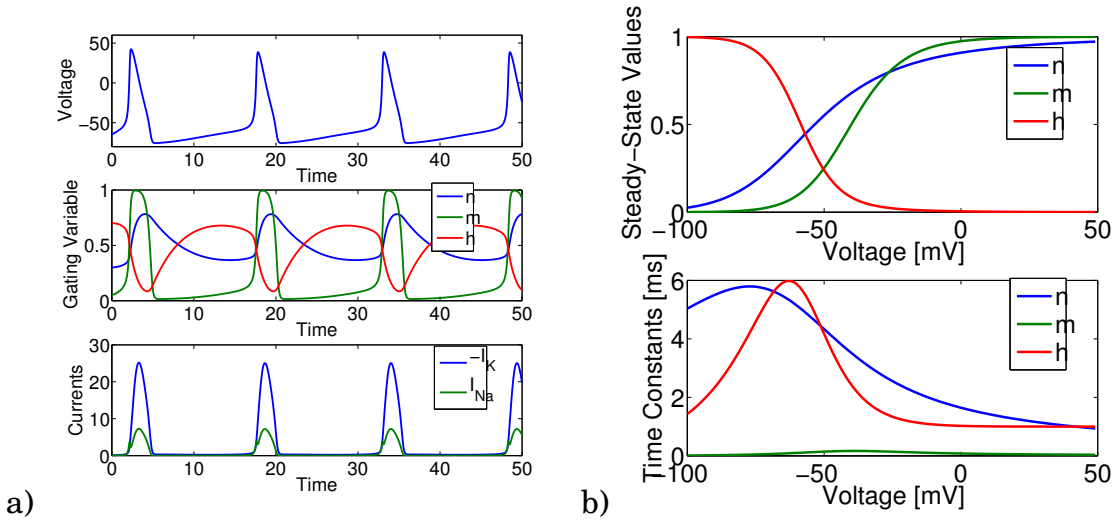



Figure 24: a) Spiking neuron with constant current injection. For ease of comparison the absolute value of the negative $Na^+$-current is plotted . b) Voltage dependence of the parameters for the gating variables.

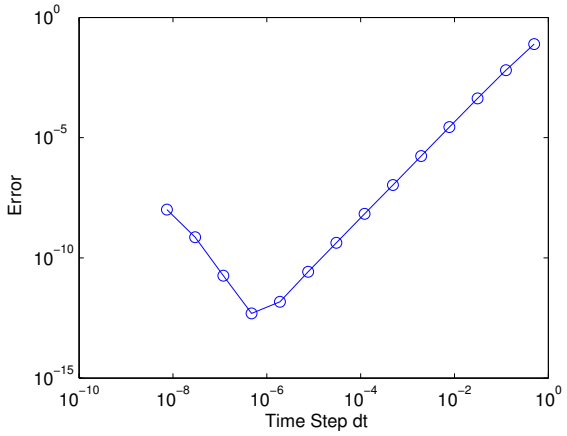



Figure 25: The round-off error leads to an increase in error with decreasing time step when $\Delta t$ is too small, $\Delta t < \Delta t_{opt}$ .

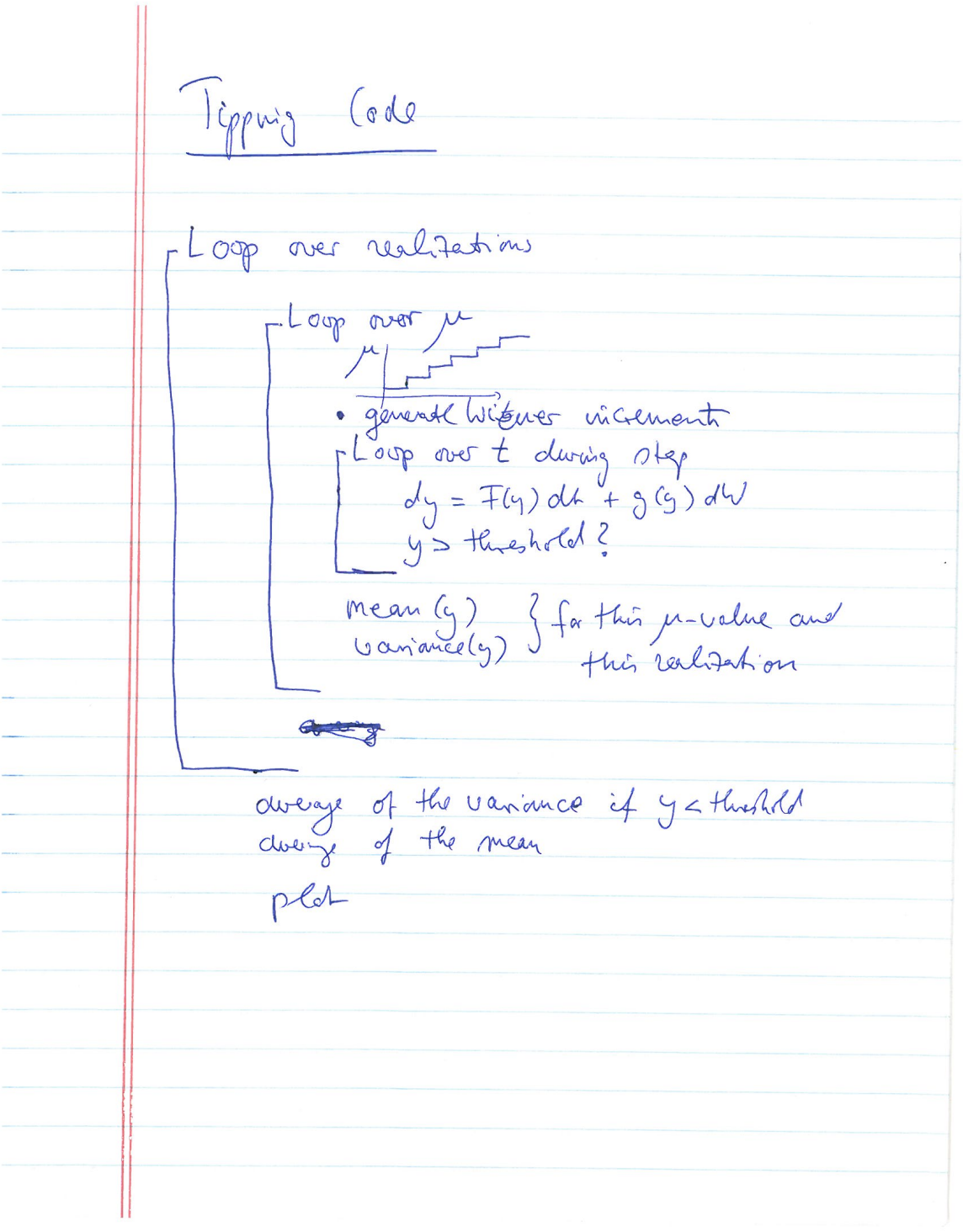



Figure 26: Pseudo-code for tipping problem.