Handling stiff ordinary differential equations (ODEs) using IMEX methods in JULIA

TKP4555 Process-System Engineering Specialization Module
TKP11 Advanced Process Simulation

Kjetil Sonerud

Department of Chemical Engineering
Norwegian University of Science and Technology (NTNU)

December 17, 2014

Agenda

Outline of the presentation

Introduction

The role of differential equations

In general, many of the practical problems in science and engineering may be formulated as differential equations. Being able to solve these correctly and efficiently are of great importance to the chemical engineer.

In practise, many PDEs may be approximated – and all higher-order ODEs may be reduced – to a system of first-order ordinary differential equations. Thus, the solution of the latter covers a wide range of applications.

We are seldom able to solve the differential equations that are of practical interest analytically, so approximating the solution through the use of *numerical methods* are often the only viable option.

This is where the issue of *stiffness* arises. Handling this through the use of so-called IMEX *methods* is the main focus of the current work, and will be further explained.

Introduction

The role of differential equations

In general, many of the practical problems in science and engineering may be formulated as differential equations. Being able to solve these correctly and efficiently are of great importance to the chemical engineer.

In practise, many PDEs may be approximated – and all higher-order ODEs may be reduced – to a system of first-order ordinary differential equations. Thus, the solution of the latter covers a wide range of applications.

We are seldom able to solve the differential equations that are of practical interest analytically, so approximating the solution through the use of *numerical methods* are often the only viable option.

This is where the issue of *stiffness* arises. Handling this through the use of so-called IMEX *methods* is the main focus of the current work, and will be further explained.

Introduction

The role of differential equations

In general, many of the practical problems in science and engineering may be formulated as differential equations. Being able to solve these correctly and efficiently are of great importance to the chemical engineer.

In practise, many PDEs may be approximated – and all higher-order ODEs may be reduced – to a system of first-order ordinary differential equations. Thus, the solution of the latter covers a wide range of applications.

We are seldom able to solve the differential equations that are of practical interest analytically, so approximating the solution through the use of *numerical methods* are often the only viable option.

This is where the issue of *stiffness* arises. Handling this through the use of so-called IMEX *methods* is the main focus of the current work, and will be further explained.

Introduction

The role of differential equations

In general, many of the practical problems in science and engineering may be formulated as differential equations. Being able to solve these correctly and efficiently are of great importance to the chemical engineer.

In practise, many PDEs may be approximated – and all higher-order ODEs may be reduced – to a system of first-order ordinary differential equations. Thus, the solution of the latter covers a wide range of applications.

We are seldom able to solve the differential equations that are of practical interest analytically, so approximating the solution through the use of *numerical methods* are often the only viable option.

This is where the issue of *stiffness* arises. Handling this through the use of so-called IMEX *methods* is the main focus of the current work, and will be further explained.

Introduction

The role of differential equations

In general, many of the practical problems in science and engineering may be formulated as differential equations. Being able to solve these correctly and efficiently are of great importance to the chemical engineer.

In practise, many PDEs may be approximated – and all higher-order ODEs may be reduced – to a system of first-order ordinary differential equations. Thus, the solution of the latter covers a wide range of applications.

We are seldom able to solve the differential equations that are of practical interest analytically, so approximating the solution through the use of *numerical methods* are often the only viable option.

This is where the issue of *stiffness* arises. Handling this through the use of so-called IMEX *methods* is the main focus of the current work, and will be further explained.

Introduction

What is a stiff differential equation?

Simply put, it is (a set of) differential equations that unfold on sufficiently different time scales. The concept of stiffness is largely a practical one – no single theoretical definition applies to all problems normally considered stiff.

> *"If a numerical method is forced to use, in a certain interval of integration, a step length which is excessively small in relation to the smoothness of the exact solution in that interval, then the problem is said to be stiff in that interval."*
> *– (Lambert, 1991)*

It is an observed fact that *implicit methods* normally outperform *explicit methods* when stiff problems arise. However, *explicit methods* are usually faster in the general case. Thus, there is a trade-off. This is where IMEX methods come in, being "the best of both worlds"

Introduction

What is a stiff differential equation?

Simply put, it is (a set of) differential equations that unfold on sufficiently different time scales. The concept of stiffness is largely a practical one – no single theoretical definition applies to all problems normally considered stiff.

> *"If a numerical method is forced to use, in a certain interval of integration, a step length which is excessively small in relation to the smoothness of the exact solution in that interval, then the problem is said to be stiff in that interval."*
> *– (Lambert, 1991)*

It is an observed fact that *implicit methods* normally outperform *explicit methods* when stiff problems arise. However, *explicit methods* are usually faster in the general case. Thus, there is a trade-off. This is where IMEX methods come in, being "the best of both worlds"

Introduction

What is a stiff differential equation?

Simply put, it is (a set of) differential equations that unfold on sufficiently different time scales. The concept of stiffness is largely a practical one – no single theoretical definition applies to all problems normally considered stiff.

> *"If a numerical method is forced to use, in a certain interval of integration, a step length which is excessively small in relation to the smoothness of the exact solution in that interval, then the problem is said to be stiff in that interval."*
> *– (Lambert, 1991)*

It is an observed fact that *implicit methods* normally outperform *explicit methods* when stiff problems arise. However, *explicit methods* are usually faster in the general case. Thus, there is a trade-off. This is where IMEX methods come in, being "the best of both worlds"

NTNU

Introduction

What is a stiff differential equation?

Simply put, it is (a set of) differential equations that unfold on sufficiently different time scales. The concept of stiffness is largely a practical one – no single theoretical definition applies to all problems normally considered stiff.

> *"If a numerical method is forced to use, in a certain interval of integration, a step length which is excessively small in relation to the smoothness of the exact solution in that interval, then the problem is said to be stiff in that interval."*
> *– (Lambert, 1991)*

It is an observed fact that *implicit methods* normally outperform *explicit methods* when stiff problems arise. However, *explicit methods* are usually faster in the general case. Thus, there is a trade-off. This is where IMEX methods come in, being "the best of both worlds"

Introduction

An illustrative example

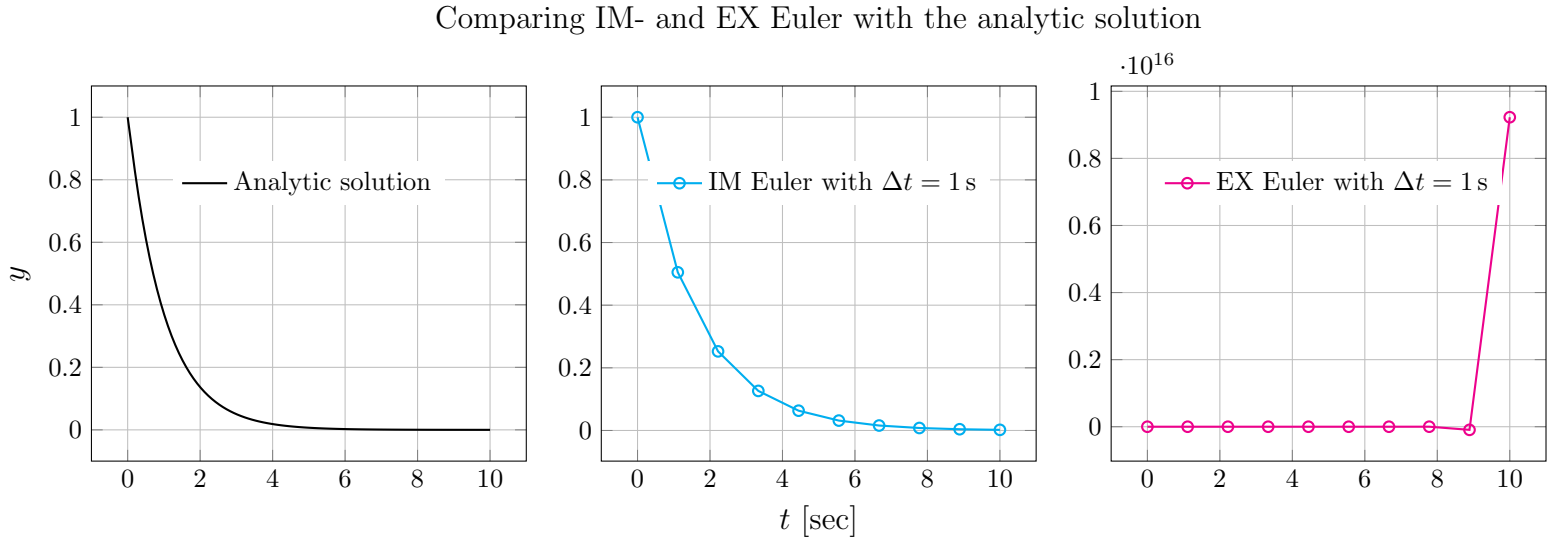


Figure: Comparison of the analytic solution to those of EX Euler and IM Euler method for the ODE-system described in $y'' + 101y' + 100y = 0$. The time step is $\Delta t = 1\,\mathrm{s}$. Note that the IM Euler solution is quite close to the analytical, but that the EX Euler method fails spectacularly.

NTNU

Introduction

An illustrative example

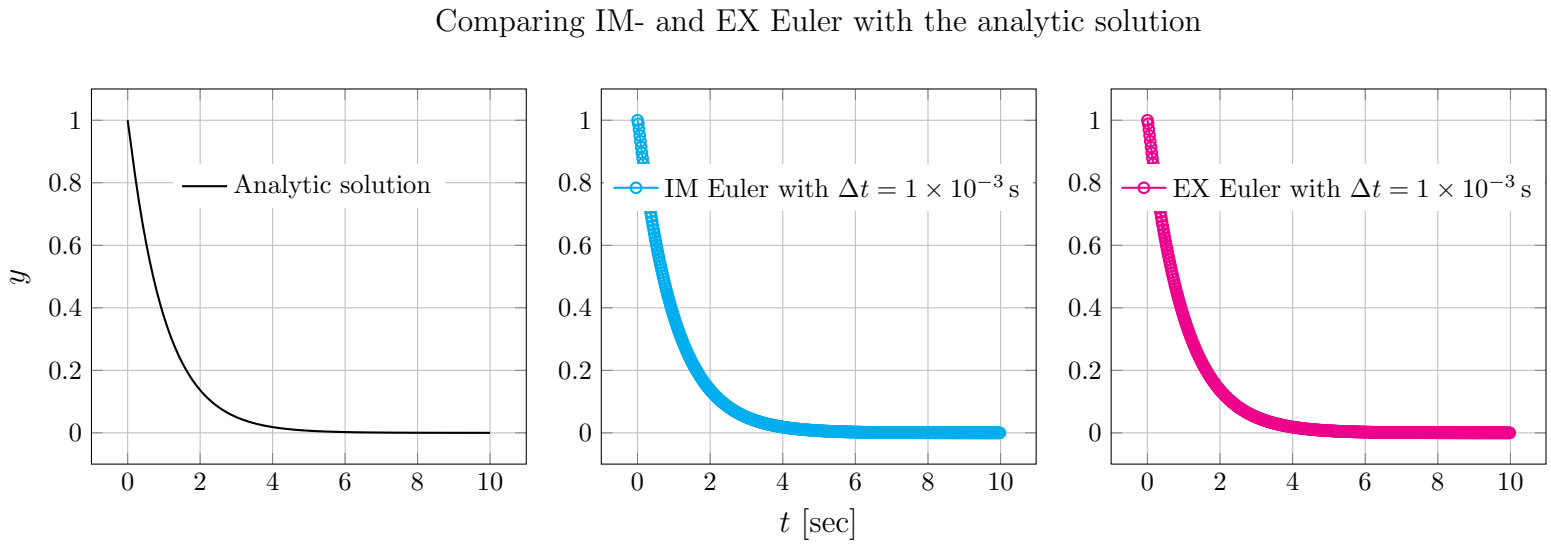


Figure: Comparison of the analytic solution to those of EX Euler and IM Euler method for the ODE-system described in $y'' + 101y' + 100y = 0$. The time step is $\Delta t = 1 \times 10^{-3}$ s. Now, both EX Euler and IM Euler are very close to the actual analytic solution.

What is an IMEX solver?

In many practical applications, large systems of ODEs that contain both stiff and non-stiff elements must be solved. Two straight-forward approaches are:

- Use an explicit method with a short time step
- Solve the whole system using an implicit scheme

The basic premise of an IMEX solver is to provide a compromise – the stiff parts of the system are solved with an implicit solver to ensure stability, while the non-stiff parts are solved with an explicit solver to reduce computational load.

$$\dot{\underline{y}}(t) = F(t, \underline{y}) + G(t, \underline{y}) \tag{1}$$

Here $F(t, \underline{y})$ represent the non-stiff part and $G(t, \underline{y})$ represent the stiff part. In general, it is assumed that *a priori* knowledge of the system is exploited to distinguish the stiff and non-stiff part so that the corresponding solver may be employed.

NTNU

What is an IMEX solver?

In many practical applications, large systems of ODEs that contain both stiff and non-stiff elements must be solved. Two straight-forward approaches are:

- Use an explicit method with a short time step
- Solve the whole system using an implicit scheme

The basic premise of an IMEX solver is to provide a compromise – the stiff parts of the system are solved with an implicit solver to ensure stability, while the non-stiff parts are solved with an explicit solver to reduce computational load.

$$\dot{\underline{y}}(t) = F(t, \underline{y}) + G(t, \underline{y}) \tag{1}$$

Here $F(t, \underline{y})$ represent the non-stiff part and $G(t, \underline{y})$ represent the stiff part. In general, it is assumed that *a priori* knowledge of the system is exploited to distinguish the stiff and non-stiff part so that the corresponding solver may be employed.

NTNU

What is an IMEX solver?

In many practical applications, large systems of ODEs that contain both stiff and non-stiff elements must be solved. Two straight-forward approaches are:

- Use an explicit method with a short time step
- Solve the whole system using an implicit scheme

The basic premise of an IMEX solver is to provide a compromise – the stiff parts of the system are solved with an implicit solver to ensure stability, while the non-stiff parts are solved with an explicit solver to reduce computational load.

$$\dot{\underline{\boldsymbol{y}}}(t) = F(t, \underline{\boldsymbol{y}}) + G(t, \underline{\boldsymbol{y}}) \tag{1}$$

Here $F(t, \underline{\boldsymbol{y}})$ represent the non-stiff part and $G(t, \underline{\boldsymbol{y}})$ represent the stiff part. In general, it is assumed that *a priori* knowledge of the system is exploited to distinguish the stiff and non-stiff part so that the corresponding solver may be employed.

Time scale analysis

Alternative strategies for solving stiff equations

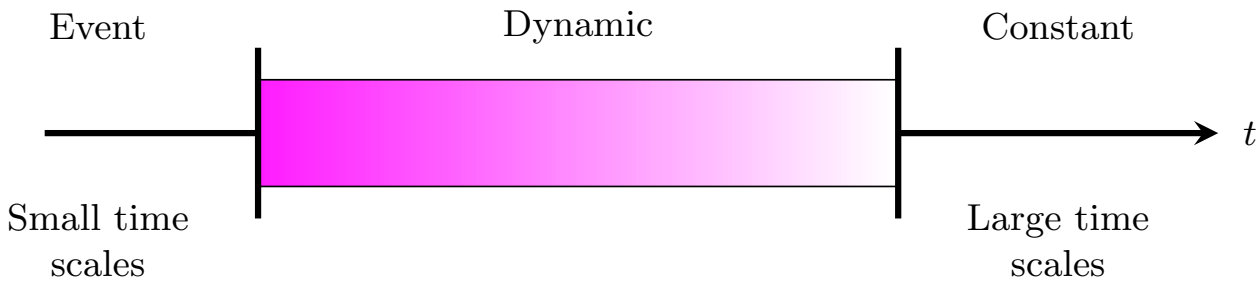


Figure: Illustration of time scales: the dynamic time scale is flanked to the left by the small time scales, which may be considered like *events*. To the right, the dynamic time scale is flanked by the large time scales, which may be considered *constant*.

Numerical solvers for ODE systems

Solving the general IVP

A general *initial value problem* (IVP) may be described as

$$\frac{\mathrm{d}\underline{\boldsymbol{y}}}{\mathrm{d}t} = \dot{\underline{\boldsymbol{y}}}(t) = f(\underline{\boldsymbol{y}}(t); \boldsymbol{\Theta}), \quad \underline{\boldsymbol{y}}(t_0) = \underline{\boldsymbol{y}}_0 \tag{2}$$

The exact solution may be found by integration of the right-hand side from t_0 to t. In general, this is not possible analytically.

Thus, a discrete approach is taken where the integral is approximated over a short interval from t_k to $t_k + \Delta_t$ by

$$\underline{\boldsymbol{y}}_{k+1} - \underline{\boldsymbol{y}}_k = (\Delta t) \cdot \underbrace{F[\underline{\boldsymbol{y}}_k, \underline{\boldsymbol{y}}_{k+1}, f(\underline{\boldsymbol{y}}; \boldsymbol{\Theta})]}_{\text{Defines the numerical method}} \tag{3}$$

NTNU

Numerical solvers for ODE systems

Solving the general IVP

A general *initial value problem* (IVP) may be described as

$$\frac{\mathrm{d}\underline{\boldsymbol{y}}}{\mathrm{d}t} = \dot{\underline{\boldsymbol{y}}}(t) = f(\underline{\boldsymbol{y}}(t); \boldsymbol{\Theta}), \quad \underline{\boldsymbol{y}}(t_0) = \underline{\boldsymbol{y}}_0 \tag{2}$$

The exact solution may be found by integration of the right-hand side from t_0 to t. In general, this is not possible analytically.

Thus, a discrete approach is taken where the integral is approximated over a short interval from t_k to $t_k + \Delta_t$ by

$$\underline{\boldsymbol{y}}_{k+1} - \underline{\boldsymbol{y}}_k = (\Delta t) \cdot \underbrace{F[\underline{\boldsymbol{y}}_k, \underline{\boldsymbol{y}}_{k+1}, f(\underline{\boldsymbol{y}}; \boldsymbol{\Theta})]}_{\text{Defines the numerical method}} \tag{3}$$

NTNU

Numerical solvers for ODE systems

Solving the general IVP

A general *initial value problem* (IVP) may be described as

$$\frac{\mathrm{d}\underline{\boldsymbol{y}}}{\mathrm{d}t} = \dot{\underline{\boldsymbol{y}}}(t) = f(\underline{\boldsymbol{y}}(t); \boldsymbol{\Theta}), \quad \underline{\boldsymbol{y}}(t_0) = \underline{\boldsymbol{y}}_0 \tag{2}$$

The exact solution may be found by integration of the right-hand side from t_0 to t. In general, this is not possible analytically.

Thus, a discrete approach is taken where the integral is approximated over a short interval from t_k to $t_k + \Delta_t$ by

$$\underline{\boldsymbol{y}}_{k+1} - \underline{\boldsymbol{y}}_k = (\Delta t) \cdot \underbrace{F[\underline{\boldsymbol{y}}_k, \underline{\boldsymbol{y}}_{k+1}, f(\underline{\boldsymbol{y}}; \boldsymbol{\Theta})]}_{\text{Defines the numerical method}} \tag{3}$$

NTNU

Numerical solvers for ODE systems

Solving the general IVP

A general *initial value problem* (IVP) may be described as

$$\frac{\mathrm{d}\underline{\boldsymbol{y}}}{\mathrm{d}t} = \dot{\underline{\boldsymbol{y}}}(t) = f(\underline{\boldsymbol{y}}(t); \boldsymbol{\Theta}), \quad \underline{\boldsymbol{y}}(t_0) = \underline{\boldsymbol{y}}_0 \tag{2}$$

The exact solution may be found by integration of the right-hand side from t_0 to t. In general, this is not possible analytically.

Thus, a discrete approach is taken where the integral is approximated over a short interval from t_k to $t_k + \Delta_t$ by

$$\underline{\boldsymbol{y}}_{k+1} - \underline{\boldsymbol{y}}_k = (\Delta t) \cdot \underbrace{F[\underline{\boldsymbol{y}}_k, \underline{\boldsymbol{y}}_{k+1}, f(\underline{\boldsymbol{y}}; \boldsymbol{\Theta})]}_{\text{Defines the numerical method}} \tag{3}$$

NTNU

Numerical solvers for ODE systems

EX and IM Euler schemes

The *explicit Euler scheme* (EX Euler, also known as "forward Euler") may be found as

$$\underline{\boldsymbol{y}}_{k+1} = \underline{\boldsymbol{y}}_k + (\Delta t) f(\underline{\boldsymbol{y}}_k; \boldsymbol{\Theta}) \tag{4}$$

The *implicit Euler scheme* (IM Euler, also known as "backward Euler") may be found as

$$\underline{\boldsymbol{y}}_{k+1} = \underline{\boldsymbol{y}}_k + (\Delta t) f(\underline{\boldsymbol{y}}_{k+1}; \boldsymbol{\Theta}) \tag{5}$$

From the equation, it is evident that a (possibly non-linear) system of equations must be solved in each time step to find $\underline{\boldsymbol{y}}_{k+1}$. In the current implementation, the Newton-Raphson method is used.

NTNU

Numerical solvers for ODE systems

EX and IM Euler schemes

The *explicit Euler scheme* (EX Euler, also known as "forward Euler") may be found as

$$\underline{\boldsymbol{y}}_{k+1} = \underline{\boldsymbol{y}}_k + (\Delta t) f(\underline{\boldsymbol{y}}_k; \boldsymbol{\Theta}) \tag{4}$$

The *implicit Euler scheme* (IM Euler, also known as "backward Euler") may be found as

$$\underline{\boldsymbol{y}}_{k+1} = \underline{\boldsymbol{y}}_k + (\Delta t) f(\underline{\boldsymbol{y}}_{k+1}; \boldsymbol{\Theta}) \tag{5}$$

From the equation, it is evident that a (possibly non-linear) system of equations must be solved in each time step to find $\underline{\boldsymbol{y}}_{k+1}$. In the current implementation, the Newton-Raphson method is used.

Numerical solvers for ODE systems

EX and IM Euler schemes

The *explicit Euler scheme* (EX Euler, also known as "forward Euler") may be found as

$$\underline{\boldsymbol{y}}_{k+1} = \underline{\boldsymbol{y}}_k + (\Delta t) f(\underline{\boldsymbol{y}}_k; \boldsymbol{\Theta}) \tag{4}$$

The *implicit Euler scheme* (IM Euler, also known as "backward Euler") may be found as

$$\underline{\boldsymbol{y}}_{k+1} = \underline{\boldsymbol{y}}_k + (\Delta t) f(\underline{\boldsymbol{y}}_{k+1}; \boldsymbol{\Theta}) \tag{5}$$

From the equation, it is evident that a (possibly non-linear) system of equations must be solved in each time step to find $\underline{\boldsymbol{y}}_{k+1}$. In the current implementation, the Newton-Raphson method is used.

NTNU

Numerical solvers for ODE systems

The IMEX Euler scheme

With the assumption that the system of ODEs are readily split into an implicit part and an explicit part, the IMEX Euler scheme follows

$$\begin{bmatrix} \underline{\boldsymbol{y}}_{k+1}^{\mathrm{IM}} \\ \\ \underline{\boldsymbol{y}}_{k+1}^{\mathrm{EX}} \end{bmatrix} = \begin{bmatrix} \underline{\boldsymbol{y}}_{k}^{\mathrm{IM}} \\ \\ \underline{\boldsymbol{y}}_{k}^{\mathrm{EX}} \end{bmatrix} + \Delta t \begin{bmatrix} f(\underline{\boldsymbol{y}}_{k+1}^{\mathrm{IM}}, \underline{\boldsymbol{y}}_{k+1}^{\mathrm{EX}}; \boldsymbol{\Theta}) \\ \\ f(\underline{\boldsymbol{y}}_{k}^{\mathrm{EX}}; \boldsymbol{\Theta}) \end{bmatrix} \tag{6}$$

Note that to solve for $\underline{\boldsymbol{y}}_{k+1}^{\mathrm{IM}}$, it is necessary to first solve for $\underline{\boldsymbol{y}}_{k+1}^{\mathrm{EX}}$, as the latter is used in the solution of the former. Thus, in the Newton-Raphson iteration to solve for $\underline{\boldsymbol{y}}_{k+1}^{\mathrm{IM}}$, it is assumed that $\underline{\boldsymbol{y}}_{k+1}^{\mathrm{EX}}$ is constant during the course of the iterations.

NTNU

Numerical solvers for ODE systems

The IMEX Euler scheme

With the assumption that the system of ODEs are readily split into an implicit part and an explicit part, the IMEX Euler scheme follows

$$\begin{bmatrix} \underline{\boldsymbol{y}}_{k+1}^{\mathrm{IM}} \\ \\ \underline{\boldsymbol{y}}_{k+1}^{\mathrm{EX}} \end{bmatrix} = \begin{bmatrix} \underline{\boldsymbol{y}}_{k}^{\mathrm{IM}} \\ \\ \underline{\boldsymbol{y}}_{k}^{\mathrm{EX}} \end{bmatrix} + \Delta t \begin{bmatrix} f(\underline{\boldsymbol{y}}_{k+1}^{\mathrm{IM}}, \underline{\boldsymbol{y}}_{k+1}^{\mathrm{EX}}; \boldsymbol{\Theta}) \\ \\ f(\underline{\boldsymbol{y}}_{k}^{\mathrm{EX}}; \boldsymbol{\Theta}) \end{bmatrix} \tag{6}$$

Note that to solve for $\underline{\boldsymbol{y}}_{k+1}^{\mathrm{IM}}$, it is necessary to first solve for $\underline{\boldsymbol{y}}_{k+1}^{\mathrm{EX}}$, as the latter is used in the solution of the former. Thus, in the Newton-Raphson iteration to solve for $\underline{\boldsymbol{y}}_{k+1}^{\mathrm{IM}}$, it is assumed that $\underline{\boldsymbol{y}}_{k+1}^{\mathrm{EX}}$ is constant during the course of the iterations.

NTNU

Numerical solvers for ODE systems

The IMEX Euler scheme

With the assumption that the system of ODEs are readily split into an implicit part and an explicit part, the IMEX Euler scheme follows

$$\begin{bmatrix} \underline{\boldsymbol{y}}_{k+1}^{\mathrm{IM}} \\ \\ \underline{\boldsymbol{y}}_{k+1}^{\mathrm{EX}} \end{bmatrix} = \begin{bmatrix} \underline{\boldsymbol{y}}_{k}^{\mathrm{IM}} \\ \\ \underline{\boldsymbol{y}}_{k}^{\mathrm{EX}} \end{bmatrix} + \Delta t \begin{bmatrix} f(\underline{\boldsymbol{y}}_{k+1}^{\mathrm{IM}}, \underline{\boldsymbol{y}}_{k+1}^{\mathrm{EX}}; \boldsymbol{\Theta}) \\ \\ f(\underline{\boldsymbol{y}}_{k}^{\mathrm{EX}}; \boldsymbol{\Theta}) \end{bmatrix} \tag{6}$$

Note that to solve for $\underline{\boldsymbol{y}}_{k+1}^{\mathrm{IM}}$, it is necessary to first solve for $\underline{\boldsymbol{y}}_{k+1}^{\mathrm{EX}}$, as the latter is used in the solution of the former. Thus, in the Newton-Raphson iteration to solve for $\underline{\boldsymbol{y}}_{k+1}^{\mathrm{IM}}$, it is assumed that $\underline{\boldsymbol{y}}_{k+1}^{\mathrm{EX}}$ is constant during the course of the iterations.

Model

The original Robertson kinetic problem

"When the equations represent the behaviour of a system containing a number of fast and slow reactions, a forward integration of these equations becomes difficult."
– (Robertson, 1966)

The original problem as stated by Robertson is one of an autocatalytic reaction

$$\mathrm{A} \xrightarrow{0.04} \mathrm{B} \qquad \text{(slow)} \tag{7a}$$

$$\mathrm{B} + \mathrm{B} \xrightarrow{3 \cdot 10^7} \mathrm{C} + \mathrm{B} \qquad \text{(very fast)} \tag{7b}$$

$$\mathrm{B} + \mathrm{C} \xrightarrow{1 \cdot 10^4} \mathrm{A} + \mathrm{C} \qquad \text{(fast)} \tag{7c}$$

Note that the reaction rate constants differ with several orders of magnitude, which we might suspect – correctly, as it turns out – makes the system stiff.

Model

The original Robertson kinetic problem

"When the equations represent the behaviour of a system containing a number of fast and slow reactions, a forward integration of these equations becomes difficult."
– (Robertson, 1966)

The original problem as stated by Robertson is one of an autocatalytic reaction

$$\mathrm{A} \xrightarrow{0.04} \mathrm{B} \qquad \text{(slow)} \tag{7a}$$

$$\mathrm{B} + \mathrm{B} \xrightarrow{3 \cdot 10^7} \mathrm{C} + \mathrm{B} \qquad \text{(very fast)} \tag{7b}$$

$$\mathrm{B} + \mathrm{C} \xrightarrow{1 \cdot 10^4} \mathrm{A} + \mathrm{C} \qquad \text{(fast)} \tag{7c}$$

Note that the reaction rate constants differ with several orders of magnitude, which we might suspect – correctly, as it turns out – makes the system stiff.

Model

The original Robertson kinetic problem

> *"When the equations represent the behaviour of a system containing a number of fast and slow reactions, a forward integration of these equations becomes difficult."*
> *– (Robertson, 1966)*

The original problem as stated by Robertson is one of an autocatalytic reaction

$$\mathrm{A} \xrightarrow{0.04} \mathrm{B} \qquad \text{(slow)} \qquad (7a)$$

$$\mathrm{B} + \mathrm{B} \xrightarrow{3 \cdot 10^7} \mathrm{C} + \mathrm{B} \qquad \text{(very fast)} \qquad (7b)$$

$$\mathrm{B} + \mathrm{C} \xrightarrow{1 \cdot 10^4} \mathrm{A} + \mathrm{C} \qquad \text{(fast)} \qquad (7c)$$

Note that the reaction rate constants differ with several orders of magnitude, which we might suspect – correctly, as it turns out – makes the system stiff.

Model

The expanded Robertson kinetic problem

An expanded version of the original problem is proposed in order to promote solution by the IMEX method central to this work. The idea behind the reaction scheme shown is to expand the non-stiff parts of the system, thus promoting the effectiveness of the IMEX solver.

$$\mathrm{A}^{\star\star} \xrightarrow{0.03} \mathrm{A}^{\star} \quad \text{(slow)} \tag{8a}$$

$$\mathrm{A}^{\star} \xrightarrow{0.05} \mathrm{A} \quad \text{(slow)} \tag{8b}$$

$$\mathrm{A} \xrightarrow{0.04} \mathrm{B} \quad \text{(slow)} \tag{8c}$$

$$\mathrm{B} + \mathrm{B} \xrightarrow{3 \cdot 10^7} \mathrm{C} + \mathrm{B} \quad \text{(very fast)} \tag{8d}$$

$$\mathrm{B} + \mathrm{C} \xrightarrow{1 \cdot 10^4} \mathrm{A} + \mathrm{C} \quad \text{(fast)} \tag{8e}$$

$$\mathrm{C} \xrightarrow{0.05} \mathrm{C}^{\star} \quad \text{(slow)} \tag{8f}$$

$$\mathrm{C}^{\star} \xrightarrow{0.04} \mathrm{C}^{\star\star} \quad \text{(slow)} \tag{8g}$$

Model

The expanded Robertson kinetic problem

An expanded version of the original problem is proposed in order to promote solution by the IMEX method central to this work. The idea behind the reaction scheme shown is to expand the non-stiff parts of the system, thus promoting the effectiveness of the IMEX solver.

$$\mathrm{A}^{\star\star} \xrightarrow{0.03} \mathrm{A}^{\star} \qquad \text{(slow)} \qquad (8a)$$

$$\mathrm{A}^{\star} \xrightarrow{0.05} \mathrm{A} \qquad \text{(slow)} \qquad (8b)$$

$$\mathrm{A} \xrightarrow{0.04} \mathrm{B} \qquad \text{(slow)} \qquad (8c)$$

$$\mathrm{B} + \mathrm{B} \xrightarrow{3 \cdot 10^7} \mathrm{C} + \mathrm{B} \qquad \text{(very fast)} \qquad (8d)$$

$$\mathrm{B} + \mathrm{C} \xrightarrow{1 \cdot 10^4} \mathrm{A} + \mathrm{C} \qquad \text{(fast)} \qquad (8e)$$

$$\mathrm{C} \xrightarrow{0.05} \mathrm{C}^{\star} \qquad \text{(slow)} \qquad (8f)$$

$$\mathrm{C}^{\star} \xrightarrow{0.04} \mathrm{C}^{\star\star} \qquad \text{(slow)} \qquad (8g)$$

NTNU

Results

Original Robertson problem

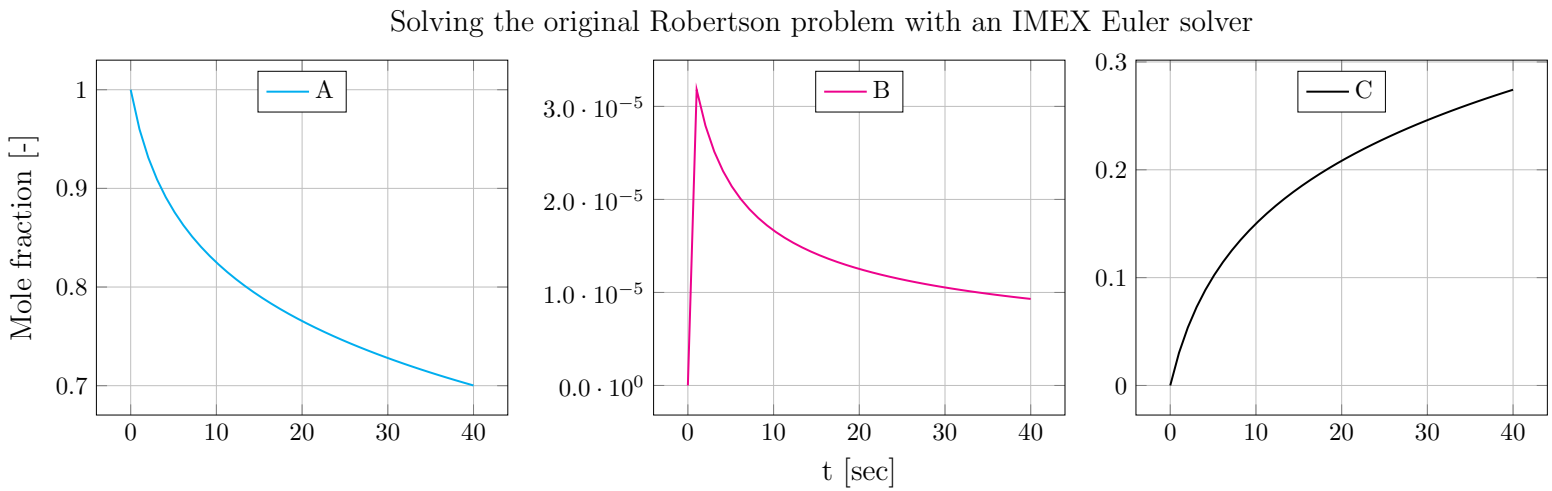


Figure: Solving the Robertson kinetics problem – original version – with IMEX Euler. The step size in this case is $\Delta t = 1\,\mathrm{s}$.

Results

Original Robertson problem

Table: Key performance data using different numerical methods for solving the original Robertson kinetics problem

Case	Δt	Number of time steps	CPU-time	Total number of N-R iterations
EX Euler (0 s to 40 s)	6×10^{-4} s	66667	0.161 s	–
IM Euler (0 s to 40 s)	1 s	40	0.0239 s	131
IMEX Euler (0 s to 40 s)	1 s	40	0.122 s	131
EX Euler (0 s to 1000 s)	3×10^{-4} s	3333333	7.789 s	–
IM Euler (0 s to 1000 s)	1 s	1000	0.0612 s	2138
IMEX Euler (0 s to 1000 s)	1 s	1000	0.0804 s	2123

Note that the IM method perform consistently better then the IMEX method, but that both these perform better than the EX method. This clearly evident for the 0 s to 1000 s simulation.

NTNU

Results

Original Robertson problem

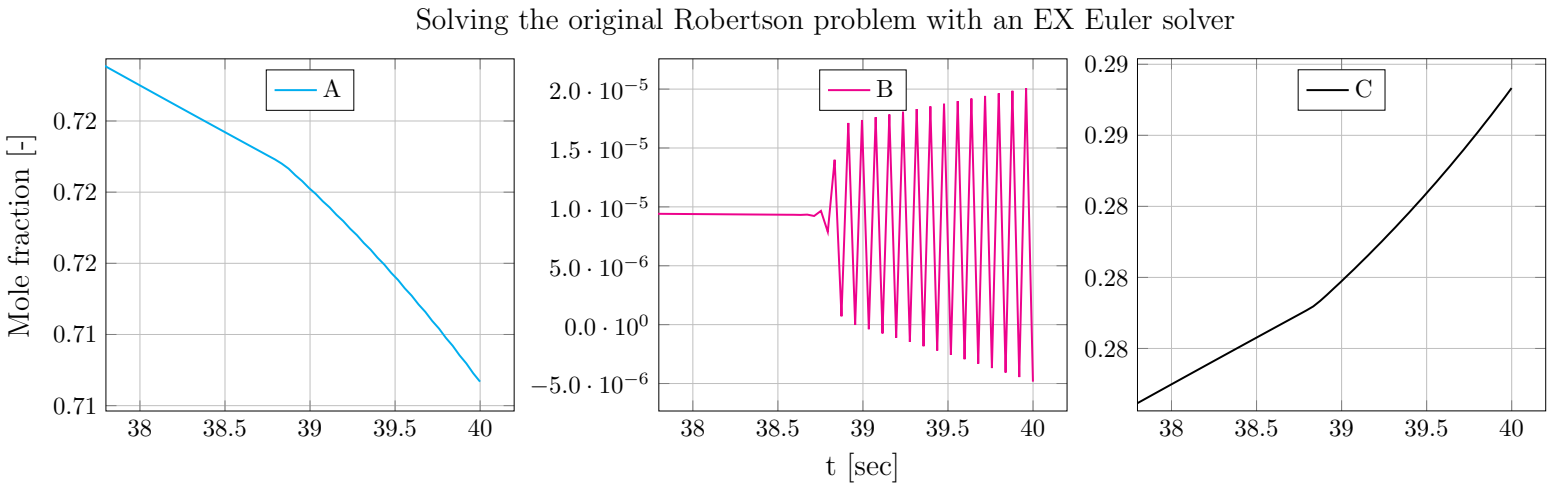


Figure: Solving the Robertson kinetics problem – original version – with EX Euler. The emerging stability problem when calculating the B-trajectory are evident.

NTNU

Results

Expanded Robertson problem

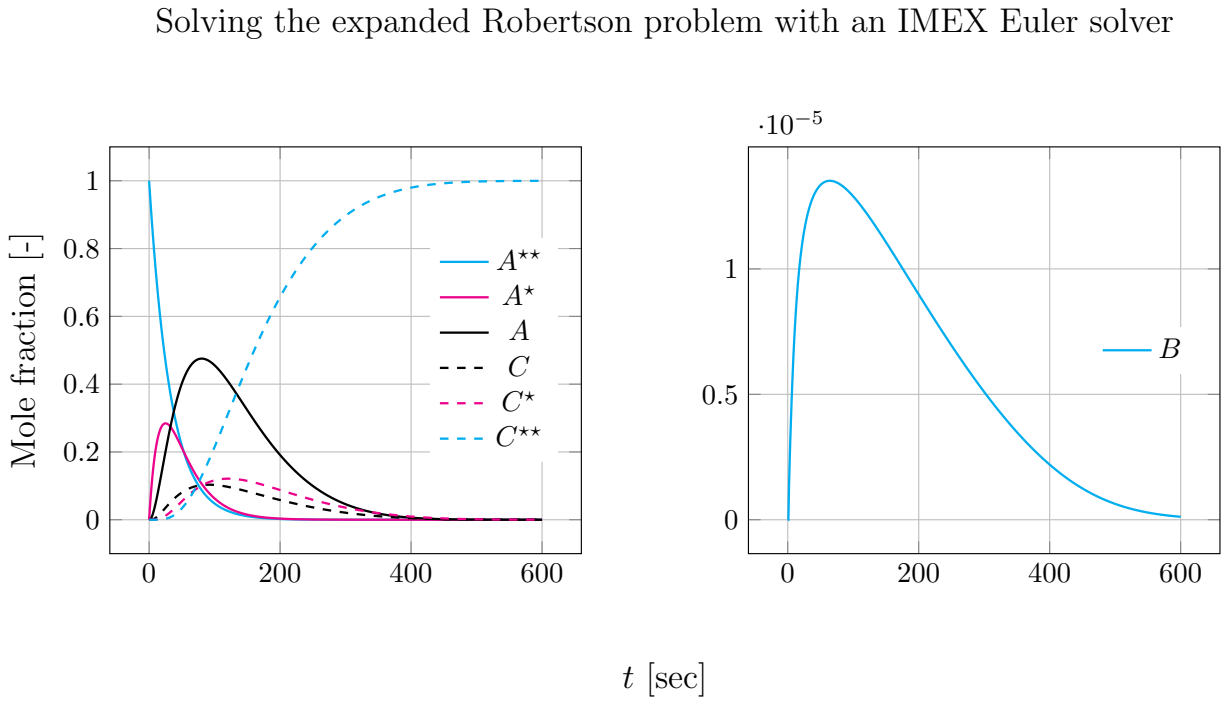


Figure: Solving the Robertson kinetics problem – expanded version – with an IMEX Euler for the time interval 0 s to 600 s. The step size in this case is $\Delta t = 1$ s.

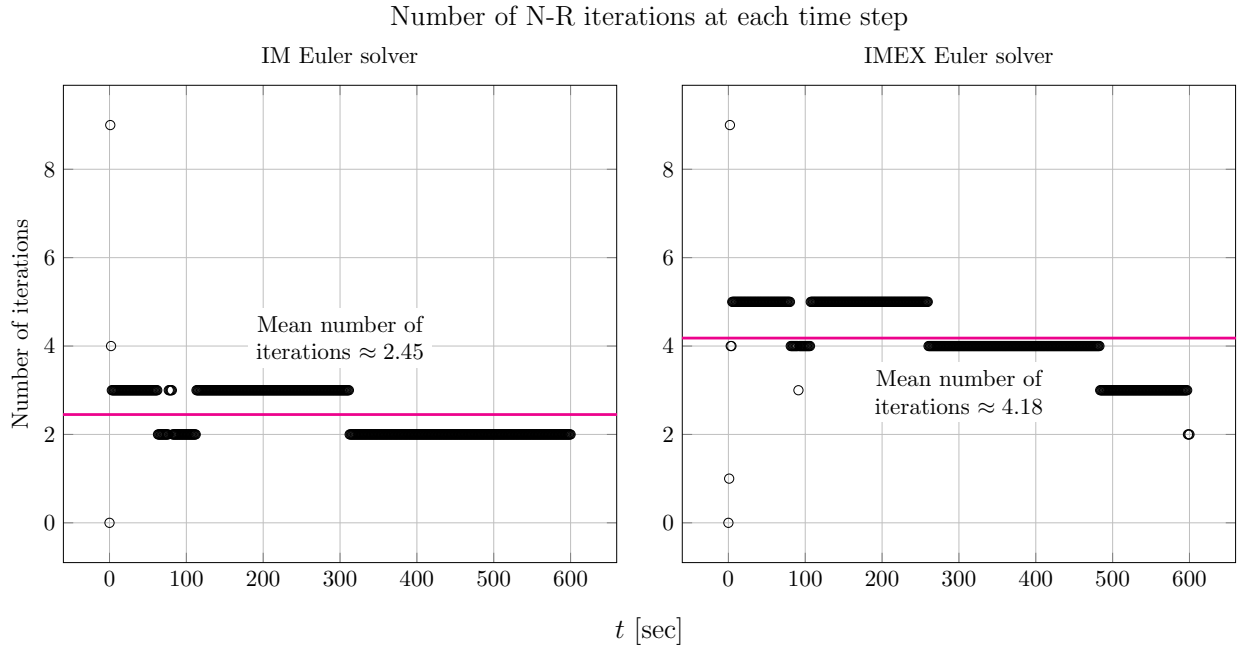
NTNU

Results

Expanded Robertson problem

Table: Key performance data using different numerical methods for solving the expanded Robertson kinetics problem

Case	Δt	Number of time steps	CPU-time	Total number of N-R iterations
EX Euler (0 s to 600 s)	1×10^{-3} s	600000	2.160 s	–
IM Euler (0 s to 600 s)	1 s	600	0.0533 s	1471
IMEX Euler (0 s to 600 s)	1 s	600	0.143 s	2506



NTNU

Discussion

From the results, it seems evident that the current IMEX Euler implementation is less effective than the pure IM Euler implementation on the current model system. This may be due to several factors:

- Sub-optimal implementation of the IMEX solver
- The system is not sufficiently large (more precisely, the non-stiff part is relatively too small) to fully exploit the IMEX scheme
- The added benefit of solving the whole system implicitly in JULIA outweighs the added cost of solving a larger system of equations

Thus, further work should be done to investigate

- Better IMEX implementations, possibly with non-Euler methods
- Larger systems of ODEs, especially ones with a small stiff and a large non-stiff part

Discussion

From the results, it seems evident that the current IMEX Euler implementation is less effective than the pure IM Euler implementation on the current model system. This may be due to several factors:

- Sub-optimal implementation of the IMEX solver
- The system is not sufficiently large (more precisely, the non-stiff part is relatively too small) to fully exploit the IMEX scheme
- The added benefit of solving the whole system implicitly in JULIA outweighs the added cost of solving a larger system of equations

Thus, further work should be done to investigate

- Better IMEX implementations, possibly with non-Euler methods
- Larger systems of ODEs, especially ones with a small stiff and a large non-stiff part

Discussion

From the results, it seems evident that the current IMEX Euler implementation is less effective than the pure IM Euler implementation on the current model system. This may be due to several factors:

- Sub-optimal implementation of the IMEX solver
- The system is not sufficiently large (more precisely, the non-stiff part is relatively too small) to fully exploit the IMEX scheme
- The added benefit of solving the whole system implicitly in JULIA outweighs the added cost of solving a larger system of equations

Thus, further work should be done to investigate

- Better IMEX implementations, possibly with non-Euler methods
- Larger systems of ODEs, especially ones with a small stiff and a large non-stiff part