1.723 - COMPUTATIONAL METHODS FOR FLOW IN POROUS MEDIA
Spring 2009

NUMERICAL METHODS FOR FIRST ORDER ODEs

Luis Cueto-Felgueroso

1. PROBLEM STATEMENT

Consider a system of first order ordinary differential equations of the form

$$\frac{d\boldsymbol{u}}{dt} = \boldsymbol{f}(t, \boldsymbol{u}), \qquad 0 \leq t \leq T, \qquad \boldsymbol{u}(t=0) = \boldsymbol{u}_0, \tag{1}$$

where $\boldsymbol{u}$ and $\boldsymbol{f}$ are vectors with N components, $\boldsymbol{u} = \boldsymbol{u}(t)$, and $\boldsymbol{f}$ is in general a nonlinear function of t and $\boldsymbol{u}$. When $\boldsymbol{f}$ does not depend explicitly on t, we say that the system (1) is autonomous. We discretize the time domain $t \in [0, T]$ as

$$0 = t^0 < t^1 < \ldots < t^n < \ldots < t^{M-1} < t^M = T, \tag{2}$$

and seek numerical methods that approximate $\boldsymbol{u}$ at at times $\{t^n, n = 0, \ldots, M\}$, in the sense that

$$\boldsymbol{u}^n \approx \boldsymbol{u}(t = t^n), \tag{3}$$

with the initial condition $\boldsymbol{u}^0 = \boldsymbol{u}(t=0) = \boldsymbol{u}_0$. The two main families of numerical methods for ODEs are one-step and multistep methods (Figure 1).

2. LINEAR MULTISTEP METHODS I. ADAMS-BASHFORTH

2.1. General form

The explicit Adams methds (Adams-Bashforth) can be written as

$$\boldsymbol{u}^{n+1} = \boldsymbol{u}^n + \Delta t \sum_{j=1}^{k} \beta_j \boldsymbol{f}^{n-j+1}, \tag{4}$$

where

$$\beta_j = (-1)^{j-1} \sum_{i=j-1}^{k-1} \binom{i}{j-1} \gamma_i, \quad \text{and} \quad \gamma_i = (-1)^i \int_0^1 \binom{-s}{i} ds. \tag{5}$$

The order of these methods is $p = k$. Some examples are given below.

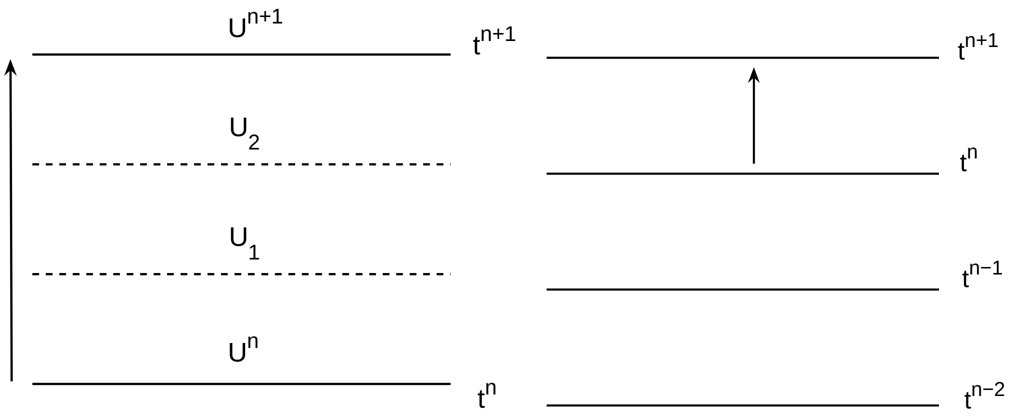

Figure 1. Schematic of solution update in one-step (left) and multistep (right) methods.

2.1.1. *k=1 (Forward Euler)*

$$\boldsymbol{u}^{n+1} = \boldsymbol{u}^n + \Delta t \boldsymbol{f}^n \tag{6}$$

2.1.2. *k=2*

$$\boldsymbol{u}^{n+1} = \boldsymbol{u}^n + \frac{\Delta t}{2}\left(3\boldsymbol{f}^n - \boldsymbol{f}^{n-1}\right) \tag{7}$$

2.1.3. *k=3*

$$\boldsymbol{u}^{n+1} = \boldsymbol{u}^n + \frac{\Delta t}{12}\left(23\boldsymbol{f}^n - 16\boldsymbol{f}^{n-1} + 5\boldsymbol{f}^{n-2}\right) \tag{8}$$

2.1.4. *k=4*

$$\boldsymbol{u}^{n+1} = \boldsymbol{u}^n + \frac{\Delta t}{24}\left(55\boldsymbol{f}^n - 59\boldsymbol{f}^{n-1} + 37\boldsymbol{f}^{n-2} - 9\boldsymbol{f}^{n-3}\right) \tag{9}$$

3. LINEAR MULTISTEP METHODS II. ADAMS-MOULTON

3.1. *General form*

The implicit Adams methods (Adams-Moulton) can be written as

$$\boldsymbol{u}^{n+1} = \boldsymbol{u}^n + \Delta t \sum_{j=0}^{k} \beta_j \boldsymbol{f}^{n-j+1}. \tag{10}$$

The order of these schemes is $p = k + 1$. Some examples are given below.

3.1.1. *k=0 (Backward Euler)*

$$\boldsymbol{u}^{n+1} = \boldsymbol{u}^n + \Delta t \boldsymbol{f}^{n+1} \tag{11}$$

3.1.2. k=1 (Trapezoidal rule)

$$\boldsymbol{u}^{n+1} = \boldsymbol{u}^n + \frac{\Delta t}{2}\left(\boldsymbol{f}^{n+1} + \boldsymbol{f}^n\right) \tag{12}$$

3.1.3. k=2

$$\boldsymbol{u}^{n+1} = \boldsymbol{u}^n + \frac{\Delta t}{12}\left(5\boldsymbol{f}^{n+1} + 8\boldsymbol{f}^n - \boldsymbol{f}^{n-1}\right) \tag{13}$$

3.1.4. k=3

$$\boldsymbol{u}^{n+1} = \boldsymbol{u}^n + \frac{\Delta t}{24}\left(9\boldsymbol{f}^{n+1} + 19\boldsymbol{f}^n - 5\boldsymbol{f}^{n-1} + \boldsymbol{f}^{n-2}\right) \tag{14}$$

3.1.5. k=4

$$\boldsymbol{u}^{n+1} = \boldsymbol{u}^n + \frac{\Delta t}{720}\left(251\boldsymbol{f}^{n+1} + 646\boldsymbol{f}^n - 264\boldsymbol{f}^{n-1} + 106\boldsymbol{f}^{n-2} - 19\boldsymbol{f}^{n-3}\right) \tag{15}$$

4. LINEAR MULTISTEP METHODS III. BACKWARD DIFFERENTIATION

The Backward Differentiation Formulas (BDF) are implicit methods, based on one-sided differences that approximate $d\boldsymbol{u}/dt$ directly. The general form is

$$\sum_{i=0}^{k} \alpha_i \boldsymbol{u}^{n-i+1} = \Delta t \beta_0 \boldsymbol{f}^{n+1}. \tag{16}$$

The order of these schemes is $p = k$. Some examples are given below.

4.0.6. BDF1, k= 1 (Backward Euler)

$$\boldsymbol{u}^{n+1} - \boldsymbol{u}^n = \Delta t \boldsymbol{f}^{n+1} \tag{17}$$

4.0.7. BDF2, k= 2

$$\boldsymbol{u}^{n+1} - \frac{4}{3}\boldsymbol{u}^n + \frac{1}{3}\boldsymbol{u}^{n-1} = \Delta t \frac{2}{3}\boldsymbol{f}^{n+1} \tag{18}$$

4.0.8. BDF3, k= 3

$$\boldsymbol{u}^{n+1} - \frac{18}{11}\boldsymbol{u}^n + \frac{9}{11}\boldsymbol{u}^{n-1} - \frac{2}{11}\boldsymbol{u}^{n-2} = \Delta t \frac{6}{11}\boldsymbol{f}^{n+1} \tag{19}$$

5. ONE-STEP METHODS: RUNGE-KUTTA METHODS

5.1. General definition

One step of an s-stage Runge-Kutta scheme can be written as

$$\boldsymbol{u}^{n+1} = \boldsymbol{u}^n + \Delta t \sum_{i=1}^{s} b_i \boldsymbol{k}_i, \qquad \boldsymbol{k}_i = \boldsymbol{f}\left(t^n + c_i \Delta t, \boldsymbol{u}_i\right), \tag{20}$$

with stage values

$$\boldsymbol{u}_i = \boldsymbol{u}^n + \Delta t \sum_{j=1}^{s} a_{ij} \boldsymbol{k}_j. \tag{21}$$

In the above expressions, Δt is the time step, and $\boldsymbol{A} = \{a_{ij}\} \in \mathbb{R}^{s \times s}$, $\boldsymbol{b} \in \mathbb{R}^s$ and $\boldsymbol{c} \in \mathbb{R}^s$ are the characteristic coefficients of each given Runge-Kutta scheme, which can be compactly written using the so-called Butcher tableau

$$\begin{array}{c|cccc}
c_1 & a_{11} & a_{12} & \cdots & a_{1s} \\
c_2 & a_{21} & a_{22} & \cdots & a_{2s} \\
\vdots & \vdots & \vdots & \ddots & \vdots \\
c_s & a_{s1} & a_{s2} & \cdots & a_{ss} \\
\hline
 & b_1 & b_2 & \cdots & b_s \\
\hline
 & \widehat{b}_1 & \widehat{b}_2 & \cdots & \widehat{b}_s
\end{array}$$

The consistency vector $\boldsymbol{c}$ defines the points (in time) at which the method computes approximations to the initial value problem, so that the stage values can be seen as $\boldsymbol{u}_i \approx \boldsymbol{u}\left(t^n + c_i \Delta t\right)$. The row sum condition

$$c_i = \sum_{j=1}^{s} a_{ij}, \quad \forall i = 1, \dots, s, \tag{22}$$

is usually adopted to simplify the order conditions for high-order methods. Explicit schemes are characterized by $\{a_{ij} = 0,\ \forall j \geq i\}$. The second set of coefficients $\{\widehat{b}_i, \quad i = 1, \dots, s\}$ corresponds to the embedded scheme, which is used for error estimation. Thus, a second approximation $\widehat{\boldsymbol{u}}^{n+1}$ can be defined using the same coefficients $\boldsymbol{A} = \{a_{ij}\} \in \mathbb{R}^{s \times s}$, and $\boldsymbol{c} \in \mathbb{R}^s$, and stage values $\boldsymbol{u}_i, i = 1, \dots, s$, as

$$\widehat{\boldsymbol{u}}^{n+1} = \boldsymbol{u}^n + \Delta t \sum_{i=1}^{s} b_i \boldsymbol{k}_i, \qquad \boldsymbol{k}_i = \boldsymbol{f}\left(t^n + c_i \Delta t, \boldsymbol{u}_i\right). \tag{23}$$

The difference between $\widehat{\boldsymbol{u}}^{n+1}$ and $\boldsymbol{u}^{n+1}$ gives an estimate of the error incurred by the numerical approximation, thus providing a criterion for time step adaptivity.

5.2. *Diagonally implicit schemes*

Among implicit RK schemes, the most popular ones for the time integration of PDEs are diagonally implicit. Their Butcher's tableaux typically take the form

$$
\begin{array}{c|ccccc}
0 & 0 & 0 & 0 & \cdots & 0 \\
2\gamma & \gamma & \gamma & 0 & \cdots & 0 \\
c_3 & a_{31} & a_{32} & \gamma & \cdots & 0 \\
\vdots & \vdots & \vdots & \vdots & \ddots & \vdots \\
1 & a_{s,1} & a_{s,2} & a_{s,3} & \cdots & \gamma \\
\hline
 & b_1 & b_2 & b_3 & \cdots & \gamma \\
\hline
 & \widehat{b}_1 & \widehat{b}_2 & \widehat{b}_3 & \cdots & \widehat{b}_s
\end{array}
$$

In particular, these schemes are referred to as *Explicit first step, Single diagonal coefficient, Diagonally Implicit Runge-Kutta* (ESDIRK) methods. Each stage value of an ESDIRK scheme is at least second-order accurate.

5.2.1. Implementation The stage value computation in a DIRK scheme reads

$$
\boldsymbol{u}_i = \boldsymbol{u}^n + \Delta t \sum_{j=1}^{i} a_{ij}\, \boldsymbol{k}_j \tag{24}
$$

Given that the $i-1$ previous $\boldsymbol{k}$'s have been previously computed, (24) can be written as

$$
\boldsymbol{u}_i = \boldsymbol{E}_i + \Delta t a_{ii}\, \boldsymbol{k}_i, \qquad \boldsymbol{E}_i = \boldsymbol{u}^n + \Delta t \sum_{j=1}^{i-1} a_{ij}\, \boldsymbol{k}_j \tag{25}
$$

The above expression is, in general, a nonlinear system of equations. The $p+1$ Newton iteration associated to (25) is given by

$$
\left(\boldsymbol{I} - \Delta t a_{ii} \frac{\partial \boldsymbol{k}_i}{\partial \boldsymbol{u}} \bigg|^{p} \right) \boldsymbol{\Delta}^p \boldsymbol{u}_i = \boldsymbol{E}_i + \Delta t a_{ii}\, \boldsymbol{k}_i^p \tag{26}
$$

where $\boldsymbol{\Delta}^p \boldsymbol{u}_i = \boldsymbol{u}_i^{p+1} - \boldsymbol{u}_i^p$.

The embedded scheme uses the same raw information as the original one, but in this case it is "processed" using the second set of weights, $\{\widehat{b}_i,\, i = 1, \ldots, s\}$. Thus, at the end of each step of the RK integrator, we have

$$
\boldsymbol{u}^{n+1} = \boldsymbol{u}^n + \Delta t \sum_{i=1}^{s} b_i \boldsymbol{k}_i \qquad \widehat{\boldsymbol{u}}^{n+1} = \boldsymbol{u}^n + \Delta t \sum_{i=1}^{s} \widehat{b}_i \boldsymbol{k}_i \tag{27}
$$

and the error estimate is given by some suitable norm of the difference between these two solutions, $r^{n+1} = ||\boldsymbol{u}^{n+1} - \widehat{\boldsymbol{u}}^{n+1}||$.

5.3. Additive Runge-Kutta schemes: implicit-explicit (IMEX)

Consider systems of ordinary differential equations that can be written in additive form as

$$\frac{d\boldsymbol{u}}{dt} = \sum_{\nu=1}^{N} \boldsymbol{f}^{[\nu]}\left(t, \boldsymbol{u}\right) \tag{28}$$

where $\boldsymbol{f}^{[1]}, \boldsymbol{f}^{[2]}, \ldots, \boldsymbol{f}^{[N]}$ denote certain terms or *components* of $\boldsymbol{f}$, whose distinctive properties are worth being taken into account separately. The above expression (28) is in principle quite loose in terms of the considerations that lead to such splitting. In general, it may be advantageous to exploit the additive structure of the system (28) when either $\boldsymbol{f}$ or the unknowns $\boldsymbol{u}$ themselves present components with significantly different time scales. In our PDE numerical solution context, the former case typically corresponds to stiff-nonstiff *a priori* decompositions of the equations, whereas the latter could apply to grid-induced stiffness. The idea behind additive schemes is to use, for each component, the integrator that best suits its particular characteristics.

In the general, N-component case, the integration of (28) can be carried out through the application of N different Runge-Kutta methods, one for each of the components. A step of an s–stage, N–part Additive (ARK_N) or Partitioned (PRK_N) Runge-Kutta scheme, defined by its generalized Butcher tableau

$$\begin{array}{c|ccc|c|ccc}
c_1 & a_{11}^{[1]} & \cdots & a_{1s}^{[1]} & \cdots & a_{11}^{[N]} & \cdots & a_{1s}^{[N]} \\
\vdots & \vdots & \ddots & \vdots & \cdots & \vdots & \ddots & \vdots \\
c_s & a_{s1}^{[1]} & \cdots & a_{ss}^{[1]} & \cdots & a_{s1}^{[N]} & \cdots & a_{ss}^{[N]} \\
\hline
 & b_1^{[1]} & \cdots & b_s^{[1]} & \cdots & b_1^{[N]} & \cdots & b_s^{[N]} \\
\hline
 & \widehat{b}_1^{[1]} & \cdots & \widehat{b}_s^{[1]} & \cdots & \widehat{b}_1^{[N]} & \cdots & \widehat{b}_s^{[N]}
\end{array}$$

is given by

$$\boldsymbol{u}^{n+1} = \boldsymbol{u}^n + \Delta t \sum_{i=1}^{s} \sum_{\nu=1}^{N} b_i^{[\nu]} \boldsymbol{f}^{[\nu]}\left(t + c_i \Delta t, \boldsymbol{u}_i\right) \tag{29}$$

with stage values

$$\boldsymbol{u}_i = \boldsymbol{u}^n + \Delta t \sum_{j=1}^{s} \sum_{\nu=1}^{N} a_{ij}^{[\nu]} \boldsymbol{f}^{[\nu]}\left(t + c_j \Delta t, \boldsymbol{u}_j\right) \tag{30}$$

where

$$c_j = \sum_{k=1}^{s} a_{jk}^{[\nu]} \quad \forall \nu = 1, \ldots, N \tag{31}$$

The Butcher coefficients $\{a_{ij}^{[\nu]}\}, \{b_i^{[\nu]}\}, \{\widehat{b}_i^{[\nu]}\},\ \nu = 1, \ldots, N$ and $\{c_i\}$ are constrained by certain accuracy and stability requirements. The order conditions of the combined scheme include those specific to each elemental method, and also certain *coupling* conditions. The growth of the number of coupling conditions for increasingly higher order and number of components N is such that the practical design of ARK_N methods has been typically restricted to $N = 2$ (ARK_2). In this latter case, the system (28) is conceptually written as

$$\frac{d\boldsymbol{u}}{dt} = \boldsymbol{f}_s\left(t, \boldsymbol{u}\right) + \boldsymbol{f}_{ns}\left(t, \boldsymbol{u}\right) \tag{32}$$

where the right hand side of (28) has been generically split into *stiff* ($\boldsymbol{f}_s$) and *nonstiff* ($\boldsymbol{f}_{ns}$) terms. Two different Runge-Kutta schemes, specifically designed and coupled, are applied to each term, and the important case in our context is the *implicit-explicit* (IMEX) approach, which acknowledges the fact that the stiff part is more efficiently dealt with by means of an *implicit* integrator, whereas the nonstiff part can be straightforwardly integrated using an *explicit* scheme. In particular, many problems of practical interest are modeled by partial differential equations whose semidiscretization can be expressed in the form of (32), where $\boldsymbol{f}_s\left(t, \boldsymbol{U}\right)$ is linear but stiff, and $\boldsymbol{f}_{ns}\left(t, \boldsymbol{U}\right)$ is nonlinear but nonstiff. The resulting system of ODE's can be very efficiently integrated using the $IMEX$ approach. The combined integrators are referred to as $IMEX\ ARK_2$ methods or, when the stiff terms are linear, *linearly implicit* Runge-Kutta schemes.

A popular family of $IMEX\ ARK_2$ schemes take the form

$$\begin{array}{c|ccccc}
0 & 0 & 0 & 0 & \cdots & 0 \\
2\gamma & 2\gamma & 0 & 0 & \cdots & 0 \\
c_3 & a_{31}^{[E]} & a_{32}^{[E]} & 0 & \cdots & 0 \\
\vdots & \vdots & \vdots & \vdots & \ddots & \vdots \\
1 & a_{s,1}^{[E]} & a_{s,2}^{[E]} & a_{s,3}^{[E]} & \cdots & 0 \\
\hline
 & b_1^{[E]} & b_2^{[E]} & b_3^{[E]} & \cdots & \gamma \\
\hline
 & \widehat{b}_1^{[E]} & \widehat{b}_2^{[E]} & \widehat{b}_3^{[E]} & \cdots & \widehat{b}_s^{[E]}
\end{array}
\qquad
\begin{array}{c|ccccc}
0 & 0 & 0 & 0 & \cdots & 0 \\
2\gamma & \gamma & \gamma & 0 & \cdots & 0 \\
c_3 & a_{31}^{[I]} & a_{32}^{[I]} & \gamma & \cdots & 0 \\
\vdots & \vdots & \vdots & \vdots & \ddots & \vdots \\
1 & a_{s,1}^{[I]} & a_{s,2}^{[I]} & a_{s,3}^{[I]} & \cdots & \gamma \\
\hline
 & b_1^{[I]} & b_2^{[I]} & b_3^{[I]} & \cdots & \gamma \\
\hline
 & \widehat{b}_1^{[I]} & \widehat{b}_2^{[I]} & \widehat{b}_3^{[I]} & \cdots & \widehat{b}_s^{[I]}
\end{array}$$

In the above expression, the superscripts $[E]$ and $[I]$ have been used in reference to the *explicit* and *implicit* components of the additive ARK_2 integrator, respectively. The stage order of the implicit integrator is two.

5.3.1. Implementation One step of an s-stage two-part additive Runge-Kutta scheme, ARK_2, defined by its Butcher coefficients ($\boldsymbol{A}^{[I]}$, $\boldsymbol{A}^{[R]}$, $\boldsymbol{b}^{[I]}$, $\boldsymbol{b}^{[E]}$, $\widehat{\boldsymbol{b}}^{[I]}$, $\widehat{\boldsymbol{b}}^{[E]}$, $\boldsymbol{c}$), is given by

$$\boldsymbol{u}^{n+1} = \boldsymbol{u}^n + \Delta t \sum_{i=1}^{s} \left(b_i^{[I]} \boldsymbol{k}_i^{[I]} + b_i^{[E]} \boldsymbol{k}_i^{[E]} \right) \tag{33}$$

where $\boldsymbol{k}_i^{[I]}$ and $\boldsymbol{k}_i^{[E]}$ are the discrete counterparts of the stiff and nonstiff operators in (32), $\boldsymbol{f}_s$ and $\boldsymbol{f}_{ns}$,

$$\boldsymbol{k}_i^{[I]} = \boldsymbol{f}_s^h\left(t_i, \boldsymbol{u}_i\right) \qquad \boldsymbol{k}_i^{[E]} = \boldsymbol{f}_{ns}^h\left(t_i, \boldsymbol{u}_i\right) \tag{34}$$

and the stage values are defined as

$$\boldsymbol{u}_i = \boldsymbol{u}^n + \Delta t \sum_{j=1}^{s} \left(a_{ij}^{[I]} \boldsymbol{k}_i^{[I]} + a_{ij}^{[E]} \boldsymbol{k}_i^{[E]} \right) \tag{35}$$

Restricting our analysis on ARK_2 pairs that use DIRK schemes for the implicit part, the above expression can be rearranged to obtain

$$\boldsymbol{u}_i = \boldsymbol{u}^n + \Delta t \sum_{j=1}^{i-1} \left(a_{ij}^{[I]} \boldsymbol{k}_j^{[I]} + a_{ij}^{[E]} \boldsymbol{k}_j^{[E]} \right) + \Delta t a_{ii}^{[I]} \boldsymbol{k}_i^{[I]} \tag{36}$$

We are interested in the linearly implicit case, for which the above expression is a linear system of equations of the form

$$\left(\boldsymbol{I} - \Delta t a_{ii}^{[I]} \boldsymbol{K} \right) \boldsymbol{u}_i = \boldsymbol{u}^n + \Delta t \sum_{j=1}^{i-1} \left(a_{ij}^{[I]} \boldsymbol{k}_j^{[I]} + a_{ij}^{[E]} \boldsymbol{k}_j^{[E]} \right) \tag{37}$$

where $\boldsymbol{k}_i^{[I]} = \boldsymbol{K} \boldsymbol{u}_i$. After solving $\boldsymbol{u}_i$ from (37), we can compute $\boldsymbol{k}_i^{[I]} = \boldsymbol{f}_s \left(t_i, \boldsymbol{u}_i \right)$, and $\boldsymbol{k}_i^{[E]} = \boldsymbol{f}_{ns} \left(t_i, \boldsymbol{u}_i \right)$.

The error estimator is constructed again in terms of the solution provided by the embedded scheme,

$$\widehat{\boldsymbol{u}}^{n+1} = \boldsymbol{u}^n + \Delta t \sum_{i=1}^{s} \left(\widehat{b}_i^{[I]} \boldsymbol{k}_i^{[I]} + \widehat{b}_i^{[E]} \boldsymbol{k}_i^{[E]} \right) \tag{38}$$

and given by some suitable norm of the difference between the original and embedded solutions, $r^{n+1} = ||\boldsymbol{u}^{n+1} - \widehat{\boldsymbol{u}}^{n+1}||$.

6. SAMPLE RUNGE-KUTTA METHODS

6.1. Heun's third-order method

$$\begin{array}{c|ccc}
0 & 0 & 0 & 0 \\
1/3 & 1/3 & 0 & 0 \\
2/3 & 0 & 2/3 & 0 \\
\hline
 & 1/4 & 0 & 3/4
\end{array}$$

6.2. Kutta's third-order method

$$\begin{array}{c|ccc}
0 & 0 & 0 & 0 \\
1/2 & 1/2 & 0 & 0 \\
1 & -1 & 2 & 0 \\
\hline
 & 1/6 & 2/3 & 1/6
\end{array}$$

6.3. Kutta's fourth-order method (associated to Simpson's first quadrature rule)

$$\begin{array}{c|cccc}
0 & 0 & 0 & 0 & 0 \\
1/2 & 1/2 & 0 & 0 & 0 \\
1/2 & 0 & 1/2 & 0 & 0 \\
1 & 0 & 0 & 1 & 0 \\
\hline
 & 1/6 & 1/3 & 1/3 & 1/6
\end{array}$$

6.4. Fourth-order scheme associated to Simpson's second quadrature rule

$$\begin{array}{c|cccc}
0 & 0 & 0 & 0 & 0 \\
1/3 & 1/3 & 0 & 0 & 0 \\
2/3 & -1/3 & 1 & 0 & 0 \\
1 & 1 & -1 & 1 & 0 \\
\hline
 & 1/8 & 3/8 & 3/8 & 1/8
\end{array}$$

6.5. *Fehlberg's method*

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

6.6. *Dormand and Prince's method*

0	0	0	0	0	0	0	0
$1/5$	$1/5$	0	0	0	0	0	0
$3/10$	3/40	$9/40$	0	0	0	0	0
$4/5$	$44/45$	$-56/15$	$32/9$	0	0	0	0
$8/9$	$19372/6561$	$-25360/2187$	$64448/6561$	$-212/729$	0	0	0
1	$9017/3168$	$-355/33$	$46732/5247$	$49/176$	$-51013/18656$	0	0
1	$35/384$	0	$500/1113$	$125/192$	$-2187/6784$	$11/84$	0
	35/384	0	$500/1113$	$125/192$	$-2187/6784$	$11/84$	0
	$5179/57600$	0	$7571/16695$	$393/640$	$-92097/339200$	$187/2100$	$1/40$

6.7. *Bijl's DIRK method*

0	0	0	0	0	0	0
$\frac{1}{2}$	$\frac{1}{4}$	$\frac{1}{4}$	0	0	0	0
$\frac{83}{250}$	$\frac{8611}{62500}$	$\frac{-1743}{31250}$	$\frac{1}{4}$	0	0	0
$\frac{31}{50}$	$\frac{5012029}{34652500}$	$\frac{-654441}{2922500}$	$\frac{174375}{388108}$	$\frac{1}{4}$	0	0
$\frac{7}{20}$	$\frac{15267082809}{155376265600}$	$\frac{-71443401}{120774400}$	$\frac{730878875}{902184768}$	$\frac{2285395}{8070912}$	$\frac{1}{4}$	0
1	$\frac{82889}{524892}$	0	$\frac{15625}{83664}$	$\frac{69875}{102672}$	$\frac{-2260}{8211}$	$\frac{1}{4}$
	$\frac{82889}{524892}$	0	$\frac{15625}{83664}$	$\frac{69875}{102672}$	$\frac{-2260}{8211}$	$\frac{1}{4}$
	$\frac{4586570599}{29645900160}$	0	$\frac{178811875}{945068544}$	$\frac{814220225}{1159782912}$	$\frac{-3700637}{11593932}$	$\frac{61727}{225920}$

7. IMPLEMENTATION OF EXPLICIT RUNGE-KUTTA METHODS

7.1. A first, non-PDE example

Consider $N>1$ dogs $\{d_j, j = 1, \ldots, N\}$, that are located at the N vertices of a polygon. At $t = 0$, each dog starts chasing its neighbor counterclockwise; i.e. $d1 \rightarrow d2 \rightarrow \cdots \rightarrow d(N-1) \rightarrow dN$. The relationship "$di$ chases $d(i+1)$", $di \rightarrow d(i+1)$, is understood in the sense that, for all $t \geq 0$, the velocity vector associated to dog i points towards dog $(i+1)$. Accordingly, the equations of motion can be written as

$$\begin{aligned} \frac{dx_i}{dt} &= u_i \\ \frac{dy_i}{dt} &= v_i \qquad \forall i = 1, 2, \ldots, N, \end{aligned}$$

with initial conditions $x_i(0) = x_i^0$ and $y_i(0) = y_i^0$. The velocity vector $\mathbf{v}_i(t) = (u_i, v_i)$ is given by

$$\begin{aligned} u_i &= c\,\frac{x_{i+1} - x_i}{r_i} \\ v_i &= c\,\frac{y_{i+1} - y_i}{r_i} \end{aligned}$$

In the above expressions, c is a characteristic dog speed (which is assumed to be constant in time and equal for all dogs), and r_i is the distance between consecutive dogs,

$$r_i = \sqrt{(x_{i+1} - x_i)^2 + (y_{i+1} - y_i)^2}$$

Assuming $c = 1$, and that the dogs are initially located along the unit circumference,

$$\begin{aligned} x_i^0 &= \cos(\theta_i) \\ y_i^0 &= \sin(\theta_i), \qquad \theta_i = \frac{\pi}{N} + \frac{2\pi}{N}(i-1), \qquad i = 1, \ldots, N, \end{aligned}$$

we may integrate the trajectories $(x(t), y(t))$ using several Runge-Kutta schemes. We will stop the computation when the smallest distance between consecutive dogs is less than $\delta = 10^{-4}$. The time steps will be chosen according to

$$\Delta t^{n+1} = t^{n+1} - t^n = \kappa \min(r_i^n)$$

where $\min(r_i^n)$ is the minimum distance between consecutive dogs at time level t^n, and κ is some constant.

The exact trajectory of the first dog is given by

$$r = e^{a(\theta - \pi/N)}, \quad \theta \in \big[\frac{\pi}{N}, \infty\big), \tag{39}$$

where

$$a = \frac{\cos\dfrac{2\pi}{N} - 1}{\sin\dfrac{2\pi}{N}}. \tag{40}$$

The trajectories of the other dogs are identical, but shifted $\dfrac{2\pi}{N}$.

This problem is solved by the code `dogsrk.m`, which can be found in the folder `example_RK`. Each step of an s-stage explicit RK method works as follows:

- For every stage, we need to compute the stage value using the $\mathbf{f}$'s evaluated at previous stages, as.

$$\boldsymbol{u}_i = \boldsymbol{u}^n + \Delta t \sum_{j=1}^{s} a_{ij}\boldsymbol{k}_j. \tag{41}$$

  Once we compute the stage value, $\boldsymbol{u}_i$, we store $\mathbf{f}(t + c_i\Delta t, \boldsymbol{u}_i)$. In the code, this corresponds to

```
for istage=1:nstage;
    accum= u0;
    for jstage= 1:istage-1;
        accum= accum + dt*ARK(istage,jstage)*F(:,jstage);
    end;

    [f,r]= odefun(t+cRK(istage)*dt,accum);
    F(:,istage)= f;
end;
```

- Once all the stage values and their associate $\mathbf{f}$'s have been computed, we advance the solution as

$$\boldsymbol{u}^{n+1} = \boldsymbol{u}^n + \Delta t \sum_{i=1}^{s} b_i\boldsymbol{k}_i, \qquad \boldsymbol{k}_i = \boldsymbol{f}\left(t^n + c_i\Delta t, \boldsymbol{u}_i\right). \tag{42}$$

  In the code, this reads

```
u= u0;
for istage= 1:nstage;
    u= u + dt*bRK(istage)*F(:,istage);
end;
```

The code may work with RK methods of orders 1 to 4 (their Butcher's tableaux are given in `ButcherT.m`. Convergence results and a sample simulation with $N = 6$ dogs are shown in Figure 2. The convergence study was carried out using the code `dogsrk_conv.m`.

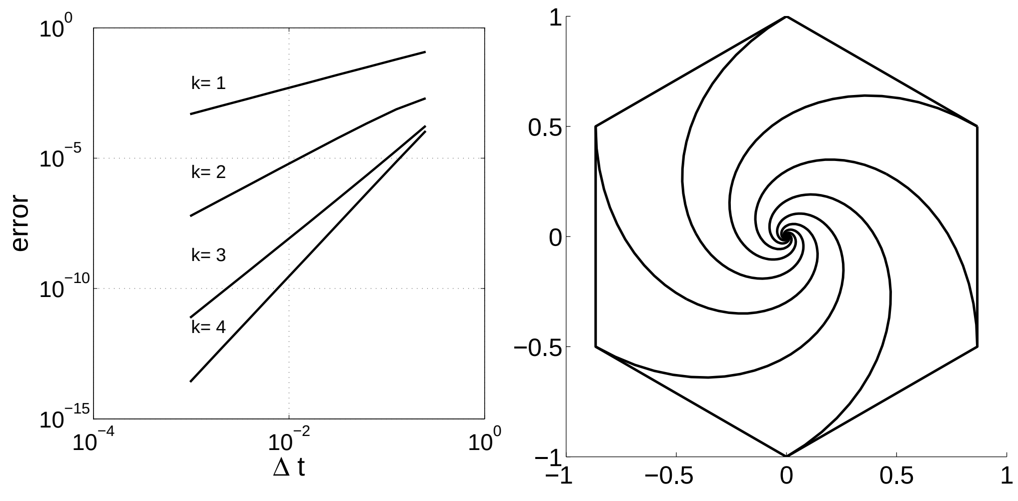

Figure 2. Left, convergence of various explicit RK methods in the chasing dogs problem. Right, trajectories for $N = 6$ dogs.

7.2. Application to PDEs

Let's solve the problem

$$\frac{\partial u}{\partial t} + \frac{\partial u}{\partial x} - \mu \frac{\partial^2 u}{\partial x^2} = 0, \qquad \mu > 0, \quad t > 0, \quad x \in [0,1], \tag{43}$$

using finite differences and explicit Runge-Kutta schemes. The initial condition is

$$u(x, t = 0) = \exp(-100(x - 0.5)^2), \tag{44}$$

and we will enforce periodic boundary conditions. This example is coded in `exp_ade.m` (folder `explicit_ADE`). We may follow the same framework of the previous example, once we consider the semi-discrete problem

$$\frac{d\boldsymbol{u}}{dt} = \boldsymbol{f}(t, \boldsymbol{u}) = \boldsymbol{K}\boldsymbol{u}, \qquad \boldsymbol{K} = -\left(\boldsymbol{D}_1 - \mu \boldsymbol{D}_2\right), \tag{45}$$

where $\boldsymbol{D}_1$ and $\boldsymbol{D}_2$ are the differentiation matrices. One step of the RK update then reads

```
for istage=1:nstage;
    accum= u0;
    for jstage= 1:istage-1;
        accum= accum + dt*ARK(istage,jstage)*F(:,jstage);
    end;

    F(:,istage)= K*accum;
end;

u= u0;
for istage= 1:nstage;
    u= u + dt*bRK(istage)*F(:,istage);
end;
```

8. IMPLEMENTATION OF IMPLICIT RUNGE-KUTTA METHODS

8.1. Linear case

In the case of (diagonally) implicit RK methods, since the evaluation of the stage values $\boldsymbol{u}_i$ involves $\boldsymbol{f}_i$ itself, we need to solve a system of equations. Let us start with the same linear advection-diffusion equation of the previous example,

$$\frac{\partial u}{\partial t} + \frac{\partial u}{\partial x} - \mu \frac{\partial^2 u}{\partial x^2} = 0, \qquad \mu > 0, \quad t > 0, \quad x \in [0,1], \tag{46}$$

initial condition

$$u(x, t=0) = \exp(-100(x-0.5)^2), \tag{47}$$

and periodic boundary conditions.

Remember that the general form of the RK update is given by

$$\boldsymbol{u}^{n+1} = \boldsymbol{u}^n + \Delta t \sum_{i=1}^{s} b_i \boldsymbol{k}_i, \qquad \boldsymbol{k}_i = \boldsymbol{f}\left(t^n + c_i \Delta t, \boldsymbol{u}_i\right), \tag{48}$$

with stage values

$$\boldsymbol{u}_i = \boldsymbol{u}^n + \Delta t \sum_{j=1}^{s} a_{ij} \boldsymbol{k}_j. \tag{49}$$

In the present case, since we consider DIRK schemes only, the stage values will be given by

$$\boldsymbol{u}_i = \boldsymbol{u}^n + \Delta t \sum_{j=1}^{i} a_{ij} \boldsymbol{k}_j, \tag{50}$$

with

$$\boldsymbol{k}_j = \boldsymbol{K}\boldsymbol{u}_j, \qquad \boldsymbol{K} = -\left(\boldsymbol{D}_1 - \mu \boldsymbol{D}_2\right), \tag{51}$$

where $\boldsymbol{D}_1$ and $\boldsymbol{D}_2$ are the differentiation matrices. Thus, the computation of stage values may be written as

$$\boldsymbol{u}_i = \boldsymbol{u}^n + \Delta t \sum_{j=1}^{i-1} a_{ij} \boldsymbol{k}_j + \Delta t a_{ii} \boldsymbol{K}\boldsymbol{u}_i. \tag{52}$$

Rearranging the above expression, we arrive at a system of linear equations of the form

$$\left(\boldsymbol{I} - \Delta t a_{ii} \boldsymbol{K}\right) \boldsymbol{u}_i = \boldsymbol{u}^n + \Delta t \sum_{j=1}^{i-1} a_{ij} \boldsymbol{k}_j, \tag{53}$$

that needs to be solved in order to get $\boldsymbol{u}_i$.

The code `imp_ade.m` (folder `implicit_ADE`) solves this problem using backward Euler and Bijl's DIRK method. One step of a DIRK scheme is coded as

```
    u0= u;
    for istage=1:nstage;
        accum= u0;
        for jstage= 1:istage-1;
            accum= accum + dt*ARK(istage,jstage)*F(:,jstage);
        end;

        %Solve system of equations
        Impmat= eye(N)-dt*ARK(istage,istage)*K;
        u= Impmat\accum;
        %Compute f(u_i)
        F(:,istage)= K*u;
    end;

    %Final update
    u= u0;
    for istage= 1:nstage;
        u= u + dt*bRK(istage)*F(:,istage);
    end;
```

Note that, in the present case, the matrix $\boldsymbol{K}$ could have been precomputed and inverted only once at the beginning.

8.2. Nonlinear case: fully implicit vs. implicit-explicit

Our model problem for the nonlinear case is the Kuramoto-Sivashinsky equation

$$\frac{\partial u}{\partial t} + \frac{\partial}{\partial x}\left(\frac{1}{2}u^2\right) + \frac{\partial^2 u}{\partial x^2} + \frac{\partial^4 u}{\partial x^4} = 0 \tag{54}$$

This equation is solved in $[0, 32\pi]$ with periodic boundary conditions. The initial condition is given by

$$u(x, t = 0) = \cos(x/16)\,(1 + \sin(x/16)) \tag{55}$$

In this equation, the low-order (advective) term is nonlinear, whereas the higher-order terms are linear. The fact that a fourth-order term is present makes the equation very stiff. There are two main strategies that could be followed: fully implicit time stepping, where the three terms are advanced implicitly, and implicit-explicit, where the advective term is advanced explicitly and the higher-order terms implicitly. The latter strategy has the advantage that we solve linear systems of equations. In the former, we need to use Newton iterations to solve the resulting nonlinear systems.

The fully nonlinear strategy is implemented in the code `imp_ks.m` (folder `implicit_KS`). One step of a DIRK scheme is implemented as

```
u0= u;
for istage=1:nstage;
    accum= u0;
    for jstage= 1:istage-1;
        accum= accum + dt*ARK(istage,jstage)*F(:,jstage);
    end;

    %Solve system of nonlinear equations
    r= 1;
    while(r>10ˆ-6);
        %Jacobian
        Jac= eye(N) + dt*ARK(istage,istage)*(D1*diag(u) + D2 + D4);
        %Residual
        R= u - accum + dt*ARK(istage,istage)*(D1*(0.5*u.*u) + D2*u + D4*u);
        %Update
        deltau= -Jac\R;
        u= u+deltau;
        r= norm(deltau);
    end;

    F(:,istage)= -(D1*(0.5*u.*u) + D2*u + D4*u);
end;

u= u0;
for istage= 1:nstage;
    u= u + dt*bRK(istage)*F(:,istage);
end;
```

The implicit-explicit strategy is implemented in the code `imex_ks.m` (folder `imex_KS`). We use one of the the ARK_2 methods developed in [6].The advantage of an IMEX formulation is that the systems that we need to solve are linear, and therefore we do not need several Newton iterations as in the fully implicit scheme. One step of the IMEX-RK scheme reads

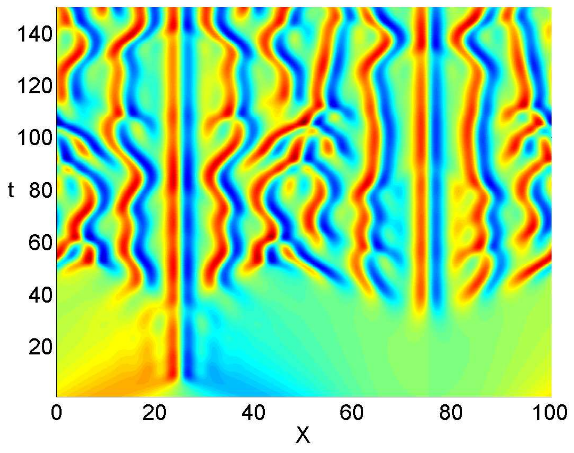

Figure 3. Schematic of solution update in one-step (left) and multistep (right) methods.

```
u0= u;
for istage=1:nstage;
    accum= u0;
    for jstage= 1:istage-1;
        accum= accum + dt*(AE(istage,jstage)*KE(:,jstage)+...
                           AI(istage,jstage)*KI(:,jstage));
    end;

    if istage>1;
        u= IKmat*accum;
    end;

    KI(:,istage)= -(D2+D4)*u;
    KE(:,istage)= -D1*(0.5*u.*u);
end;

u= u0;
for istage= 1:nstage;
    u= u + dt*( bE(istage)*KE(:,istage) + bI(istage)*KI(:,istage) );
end;
```

Since we have precomputed and inverted the matrix of the linear system of equations, the time stepping actually does not require solving a system of equations. Of course this is possible because we are using a constant time step Δt. Otherwise, we would need to recompute and invert $\boldsymbol{K}$. Figure 3 shows the simulated evolution of $u(x)$ in space-time, using the implicit-explicit code, `imex_ks.m`

9. Regions of absolute stability

9.1. Linear multistep methods

Multistep methods can be written as

$$\sum_{j=0}^{k} \alpha_j \boldsymbol{u}^{n-j+1} = \Delta t \sum_{j=0}^{k} \beta_j \boldsymbol{f}^{n-j+1}. \tag{56}$$

The boundary of the region of absolute stability of a multistep method is, based on the above definition, given by

$$z = \frac{\sum_{j=0}^{k} \alpha_j e^{i\,(-j+1)\theta}}{\sum_{j=0}^{k} \beta_j e^{i\,(-j+1)\theta}} \tag{57}$$

where $z = \lambda \Delta t$, i is the imaginary unit, and θ varies between 0 and 2π. The regions of absolute stability of several Adams and BDF methods are plotted in Figures 4 and 5, respectively, using the codes in the folder `stab_regions`.

A numerical method is A-stable if its region of absolute stability contains the left half-plane, $\mathcal{R}e(\lambda)\Delta t < 0$. It can be shown that:

- An explicit linear multistep method cannot be A-stable.
- The order of an A-stable linear multistep method cannot exceed 2.
- The second-order A-stable implicit linear multistep method with the smallest error constant is the trapezoidal method.

9.2. Runge-Kutta methods

Recall that the region of absolute stability is determined by the complex values $z = \lambda \Delta t$ for which, when the method is applied to the test equation

$$\frac{du}{dt} = \lambda u, \tag{58}$$

we get $|u^{n+1}| \leq |u^n|$. One step of an explicit Runge-Kutta method applied to the above test equation can be written as

$$u^{n+1} = \left[1 + z\boldsymbol{b}^T \left(\boldsymbol{I} - z\boldsymbol{A}\right)^{-1} \boldsymbol{1}\right] u^n. \tag{59}$$

Expanding the inverse operator, we can rewrite the above expression as

$$u^{n+1} = \left[1 + z\boldsymbol{b}^T \left(\boldsymbol{I} + z\boldsymbol{A} + \dots + z^k \boldsymbol{A}^k + \dots\right) \boldsymbol{1}\right] u^n, \tag{60}$$

or

$$u^{n+1} = R(z) u^n. \tag{61}$$

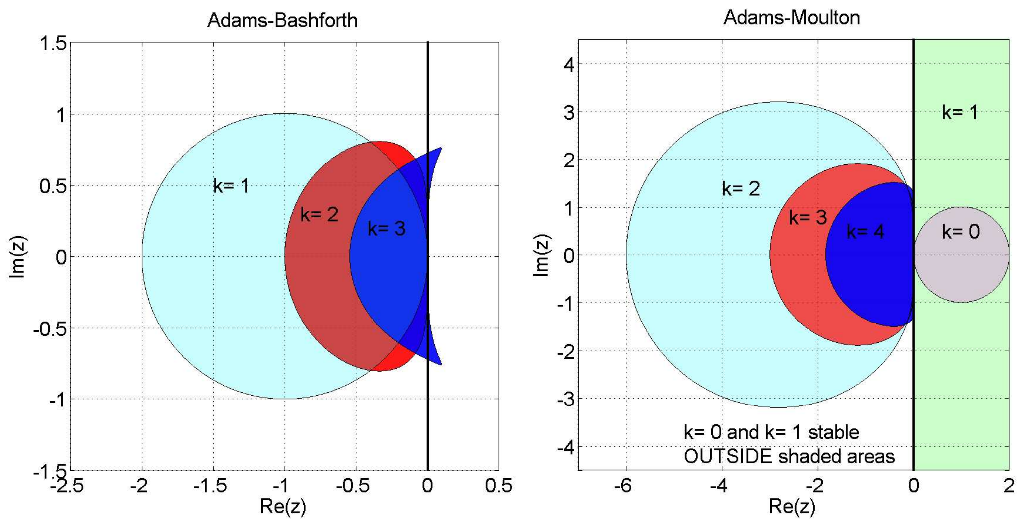

Figure 4. Regions of absolute stability of Adams methods. Left, Adams-Bashforth. Right, Adams-Moulton.

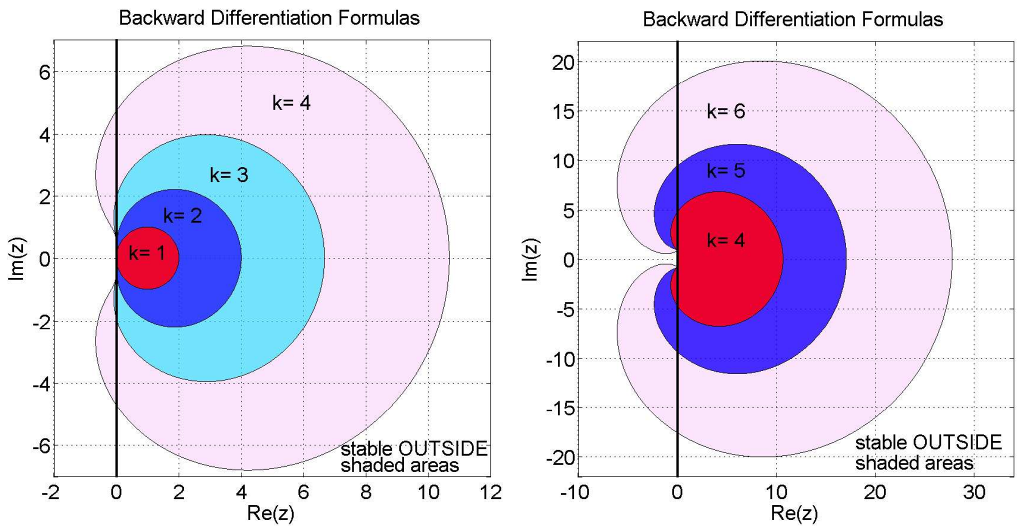

Figure 5. Regions of absolute stability of backward differentiation formulas. Left, $k = 1 - 4$. Right, $k = 4 - 6$.

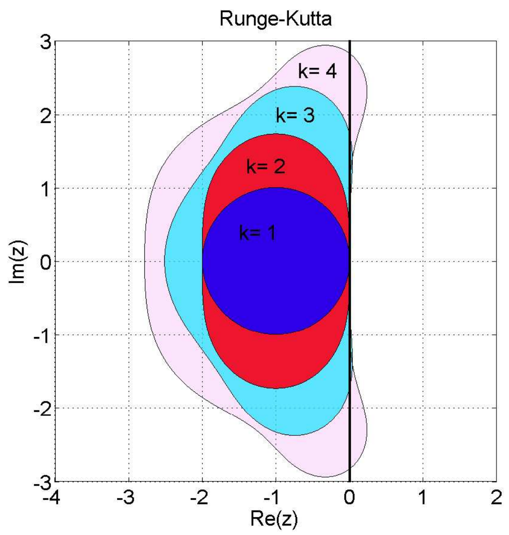

Figure 6. Regions of absolute stability of explicit Runge-Kutta methods with $p = s \leq 4$.

For a Runge-Kutta method of order p, we get

$$R(z) = 1 + z + \frac{z^2}{2} + \cdots + \frac{z^p}{p!} + \sum_{j=p+1}^{s} z^j \boldsymbol{b}^T \boldsymbol{A}^{j-1} \mathbf{1}. \tag{62}$$

In particular, the region of absolute stability of an explicit pth-order Runge-Kutta method with s stages and order $p = s \leq 4$, is given by

$$\left|1 + z + \frac{z^2}{2} + \cdots + \frac{z^p}{p!}\right| \leq 1 \tag{63}$$

Note that all p-stage explicit RK methods of order p have the same region of absolute stability. For an s-stage method of order $p < s$ the stability region depends on the method's coefficients. The stability regions of explicit RK methods with $p = s \leq 4$ are plotted in Figure 6, using the code `regions_RK.m`.

10. Stability of the method of lines

The practical relevance of determining the region of absolute stability of a given time integration scheme is that the eigenvalues of our spatial discretization matrix, multiplied by Δt, have to fall inside that region. The codes in the folder `stab_MOL` assemble the discretization matrices for the model equation

$$\frac{\partial u}{\partial t} + \frac{\partial u}{\partial x} + \mu_2 \frac{\partial^2 u}{\partial x^2} + \mu_3 \frac{\partial^3 u}{\partial x^3} + \mu_4 \frac{\partial^4 u}{\partial x^4} = 0, \tag{64}$$

and plot the scaled eigenvalues together with the regions of absolute stability of various ODE solvers. Figure 7 shows the stability regions for the Adams-Bashforth and explicit Runge-Kutta methods,

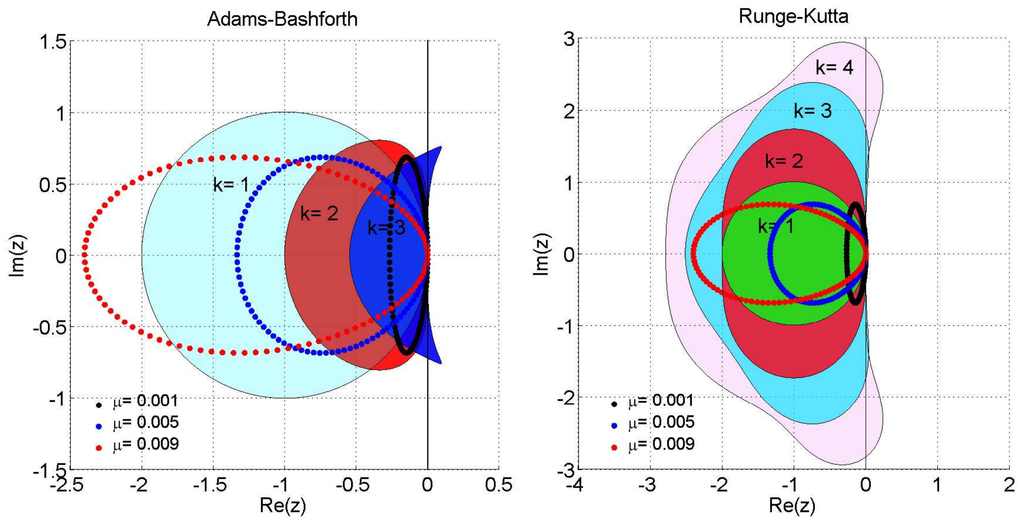

Figure 7. Stability of the method of lines. Δt-scaled eigenvalues for the model problem (65), and regions of absolute stability of common explicit schemes. Left, Adams-Basforth methods. Right, Runge-Kutta methods. The eigenvalues correspond to three different values of the diffusivity, μ.

together with the Δt-scaled eigenvalues of a finite difference discretization of the model problem

$$\frac{\partial u}{\partial t} + \frac{\partial u}{\partial x} - \mu \frac{\partial^2 u}{\partial x^2} = 0, \tag{65}$$

for several values of the diffusivity μ. We therefore compute the eigenvalues $\{\lambda_j\}$ of the discretization matrix $\boldsymbol{K}$,

$$\boldsymbol{K} = -\boldsymbol{D}_1 + \mu \boldsymbol{D}_2, \tag{66}$$

and multiply them by Δt. As μ increases, the scaled eigenvalues advance quickly inside the left half-plane, which in practice means that we need small time steps (the problem becomes stiff). In that situation, A-stable methods provide significant advantages, since the time step can be chosen based on accuracy requirements, rather than based on stability restrictions.

BIBLIOGRAPHY

1. Hairer E., Nørsett S.P., Wanner G. Solving ordinary differential equations (vols. 1 and 2). Springer-Verlag, Berlin, (1993)
2. Butcher J.C. Numerical methods for ordinary differential equations. John Wiley & Sons, Chichester, (2008)
3. Ascher U.M., Petzold L.R. Computer methods for ordinary differential equations and differential-algebraic equations. SIAM, Philadelphia, (1998)
4. Araújo A.L., Murúa A., Sanz-Serna J.M. Symplectic methods based on decompositions. *SIAM J. Numer. Anal.* **34**, 1926–1947, (1997)
5. Ascher U.M., Ruuth S.J., Spiteri R.J. Implicit-explicit Runge-Kutta methods for time-dependent partial differential equations. *Appl. Numer. Math.* **25**, 151–167, (1997)
6. Kennedy C.A., Carpenter M.H. Additive Runge-Kutta schemes for convection-diffusion-reaction equations. *Appl. Numer. Math.* **44**, 139–181, (2003)