Take-home Exam 02686
Scientific Computing for Differential Equations
Spring 2021

John Bagterp Jørgensen

April 12, 2021

You can work on the problems in groups, but you have to hand in an individual report that you write yourself. The deadline for handing in the report is June 7, 2021, 13:30. The report must be uploaded to CampusNet and a printed copy of the pdf must be handed in to my mailbox in Office 303B-112 (provided that DTU is open at that time). To CampusNet you must upload 1) a pdf of the report; 2) a zip file with a) all latex/word code used to generate the pdf; b) all matlab code used to generate the results. The grading will be based on this report.

1 Test equation for ODEs

Consider the test equation

$$\dot{x}(t) = \lambda x(t), \quad x(0) = x_0 \tag{1}$$

for $\lambda = -1$ and $x_0 = 1$.

1. Provide the analytical solution to the test equation
2. Explain and provide definitions of the local and the global truncation error
3. Compute the local and global truncation errors for the test equation when solved with a) the explicit Euler method (fixed step size) and b) the implicit Euler method (fixed step size)
4. Plot the local error vs the time step for the explicit Euler method and the implicit Euler method. Does the plot behave as you would expect. Explain what we mean by order. What is the order of the explicit Euler method and the implicit Euler method, respectively. You should base your answer on your numerical simulations. Is this as you would expect using asymptotic theoretical considerations?
5. Plot the global error vs the time step for the explicit Euler method and the implicit Euler method. Does the plot behave as you would expect.
6. Explain stability of a method and derive the expressions for the stability of the explicit Euler method and the implicit Euler method. Plot the stability regions. Explain A-stability. Is the explicit Euler-method A-stable? Is the implicit Euler method A-stable?

2 Explicit ODE solver

Consider the initial value problem

$$\dot{x}(t) = f(t, x(t), p), \quad x(t_0) = x_0, \tag{2}$$

where $x \in \mathbb{R}^{n_x}$ and $p \in \mathbb{R}^{n_p}$.

1. Describe the explicit Euler algorithm (i.e. provide an algorithm for it in your report and explain how you get from the differential equations to the numerical formulas)
2. Implement an algorithm in Matlab for the explicit Euler method with fixed time-step and provide this in your report. Use a format that enables syntax highlighting.
3. Implement an algorithm in Matlab for the explicit Euler method with adaptive time step and error estimation using step doubling
4. Test your algorithms on the van der Pol problem ($\mu = 1.5$ and $\mu = 15$, $x_0 = [1.0; 1.0]$)
5. Test your algorithms on the adiabatic CSTR problem described in the papers uploaded to Learn (3D-version and 1D-version).
6. Compare the results from your algorithms with the results you get using some of Matlab's ODE solvers

The report should contain figures and a discussion of your algorithm for different tolerances and step sizes. Discuss the interfaces (the way you call) for the ODE solver that you made.

3 Implicit ODE solver

Consider the initial value problem

$$\dot{x}(t) = f(t, x(t), p), \quad x(t_0) = x_0, \tag{3}$$

where $x \in \mathbb{R}^{n_x}$ and $p \in \mathbb{R}^{n_p}$.

1. Describe the implicit Euler algorithm (i.e. provide an algorithm for it in your report and explain how you get from the differential equations to the numerical formulas)
2. Implement an algorithm in Matlab for the implicit Euler method with fixed time-step and provide this in your report. Use a format that enables syntax highlighting.
3. Implement an algorithm in Matlab for the implicit Euler method with adaptive time step and error estimation using step doubling
4. Test your algorithms on the van der Pol problem ($\mu = 2$ and $\mu = 12$, $x_0 = [0.5; 0.5]$)
5. Test your algorithms on the adiabatic CSTR problem described in the papers uploaded to Learn (3D-version and 1D-version).
6. Compare the results from your algorithms with the results you get using some of Matlab's ODE solvers The report should contain figures and a discussion of your algorithm for different tolerances and step sizes.
7. Discuss when one should use the implicit Euler method rather than the explicit Euler method and illustrate that using an example.

Discuss the interfaces (the way you call) for the ODE solver that you made.

4 Solvers for SDEs

We consider now stochastic differential equations in the form

$$dx(t) = f(t, x(t), p_f)dt + g(t, x(t), p_g)d\omega(t) \qquad d\omega(t) \sim N_{iid}(0, Idt) \tag{4}$$

where $x \in \mathbb{R}^{n_x}$ and ω is a stochastic variable with dimension n_w. p_f and p_g are parameters for $f : \mathbb{R} \times \mathbb{R}^{n_x} \times \mathbb{R}^{n_{p_f}} \mapsto \mathbb{R}^{n_x}$ and $g : \mathbb{R} \times \mathbb{R}^{n_x} \times \mathbb{R}^{n_{p_g}} \mapsto \mathbb{R}^{n_x \times n_w}$ (i.e. the result of g is a matrix of size $n_x \times n_w$).

1. Make a function in Matlab that can realize a multivariate standard Wiener process
2. Implement the explicit-explicit method with fixed step size
3. Implement the implicit-explicit method with fixed step size
4. Describe your implementations and the idea you use in deriving the methods. Provide matlab code for your implementations.
5. Test your implementations using SDE versions of the Van der Pol problem
6. Test your algorithms on the adiabatic CSTR problem described in the papers uploaded to Learn (3D-version and 1D-version).

5 Classical Runge-Kutta method with fixed time step size

Consider the initial value problem

$$\dot{x}(t) = f(t, x(t), p), \quad x(t_0) = x_0, \tag{5}$$

where $x \in \mathbb{R}^{n_x}$ and $p \in \mathbb{R}^{n_p}$.

1. Describe the classical Runge-Kutta method with fixed step size.
2. Implement an algorithm in Matlab for the classical Runge-Kutta method with fixed time step size. Provide the code in your report. Use a format that enables syntax highlighting. Comment on the code.
3. Test your problem for test equation. Discuss order and stability of the numerical method.
4. Test your algorithms on the van der Pol problem ($\mu = 1.5$ and $\mu = 15$, $x_0 = [1.0; 1.0]$)
5. Test your algorithms on the adiabatic CSTR problem described in the papers uploaded to learn (3D-version and 1D-version).
6. Compare the results from your algorithms with the results you get using some of Matlab's ODE solvers

The report should contain figures and a discussion of your algorithm for different tolerances and step sizes. Discuss the interfaces (the way you call) for the ODE solver that you made.

6 Classical Runge-Kutta method with adaptive time step

Consider the initial value problem

$$\dot{x}(t) = f(t, x(t), p), \quad x(t_0) = x_0, \tag{6}$$

where $x \in \mathbb{R}^{n_x}$ and $p \in \mathbb{R}^{n_p}$.

1. Describe the classical Runge-Kutta method with adaptive time step size.
2. Implement an algorithm in Matlab for the classical Runge-Kutta method with adaptive time step size. Provide the code in your report. Use a format that enables syntax highlighting. Comment on the code.
3. Test your problem for test equation. Discuss order and stability of the numerical method.
4. Test your algorithms on the van der Pol problem ($\mu = 1.5$ and $\mu = 15$, $x_0 = [1.0; 1.0]$)
5. Test your algorithms on the adiabatic CSTR problem described in the papers uploaded to Learn (3D-version and 1D-version).

6. Compare the results from your algorithms with the results you get using some of Matlab's ODE solvers

The report should contain figures and a discussion of your algorithm for different tolerances and step sizes. Discuss the interfaces (the way you call) for the ODE solver that you made.

7 Dormand-Prince 5(4)

Consider the initial value problem

$$\dot{x}(t) = f(t, x(t), p), \quad x(t_0) = x_0, \tag{7}$$

where $x \in \mathbb{R}^{n_x}$ and $p \in \mathbb{R}^{n_p}$.

1. Describe the Dormand-Prince method (DOPRI54) method with adaptive time step size.
2. Implement an algorithm in Matlab for the DOPRI54 method with adaptive time step size. Provide the code in your report. Use a format that enables syntax highlighting. Comment on the code.
3. Test your problem for test equation. Discuss order and stability of the numerical method.
4. Test your algorithms on the van der Pol problem ($\mu = 1.5$ and $\mu = 15$, $x_0 = [1.0; 1.0]$)
5. Test your algorithms on the adiabatic CSTR problem described in the papers uploaded to Learn (3D-version and 1D-version).
6. Compare the results from your algorithms with the results you get using some of Matlab's ODE solvers (in particular ode45 which implements the DOPRI54 method)

8 ESDIRK23

1. Using the order conditions and other conditions derive the ESDIRK23 method.
2. Plot the stability region of the ESDIRK23 method. Is it A-stable? Is it L-stable? Discuss the practical implications of the stability region of ESDIRK23.
3. Implement ESDIRK23 with variable step size.
4. Test your algorithms on the van der Pol problem ($\mu = 1.5$ and $\mu = 15$, $x_0 = [1.0; 1.0]$)
5. Test your algorithms on the adiabatic CSTR problem described in the attached papers (3D-version and 1D-version).
6. Compare the solution and the number of function evaluations with your own explicit Runge-Kutta method to the other solvers that you have used in this exam problem. Discuss when it is appropriate to use ESDIRK23 and illustrate that with an example you choose.

9 Discussion and Conclusions

Discuss and compare the different solution methodes for the test problem used (Van der Pol and CSTR).