The Ministry of Education and Science of the Russian Federation

Lobachevsky State University of Nizhni Novgorod

Computing Mathematics and Cybernetics faculty

The competitiveness enhancement program

of the Lobachevsky State University of Nizhni Novgorod

among the world's research and education centers

Strategic initiative

“Achieving leading positions in the field of supercomputer technology

and high-performance computing”

NUMERICAL METHODS FOR SOLVING DIFFERENTIAL EQUATIONS

Lecture 1. Numerical Methods for Solving Systems of Ordinary Differential Equations

Nizhni Novgorod

2014

OBJECTIVES

The purpose of this lecture is to study numerical methods of solving systems of ordinary differential equations and approaches to their parallelization.

ABSTRACT

This lecture is dedicated to solving systems of ordinary differential equations using the Adams, Euler and Runge–Kutta methods. It discusses the issues of local error control. The lecture also describes the numerical methods to solve systems of ordinary differential equations and the method of partial discretization of partial differential equations. It also describes approaches to parallelization of methods to solve ODE systems.

GUIDELINES

This lecture is dedicated to numerical methods of solving systems of ordinary differential equations (ODE). Differential equations (both ordinary and partial) are often used to describe various phenomena and processes around us. Motion path computation for planets and Earth satellites, determination of structural characteristics of engineering facilities and weather forecasting - these are only a few problems whose mathematical models are formalized in differential equations. Although they look simple, only some special ODE types can be solved analytically.

The introductory part of this lecture defines the following notions: ODE, ODE solution, initial value and the initial value problem. Numerical methods to solve initial value problems like $u' = f(x,u)$, $x \in [a,b]$, $u_0 = u(x_0), x_0 = a$ described in this lecture are intended to obtain a numerical table of approximate values v_i of a desired solution *u(x)*at the point $x_i \in [x_0,b]$.

The most basic method of ODE numerical solution is the Euler method. Use the step size $h = (b - x_0)/n$ and set $x_i = x_0 + ih$, i=0,1,…,*n* as mesh points. The Euler method formula can be derived by representing the *u(x)* function as its Taylor series. The local truncation error of Euler methodis $O(h^2)$ while its global truncation error is $O(h)$. The lecture derives a modified Euler method formula whose global truncation error is $O(h^2)$. The method disadvantage, similar to other high order methods based on *u(x)* representation as a Taylor series and successive differentiation of the equation to obtain Taylor coefficients, is the necessity to compute partial derivatives of *f(x,u)* at each step. This operation may be quite difficult; plus, high-order methods are usually have more computational effort than simpler algorithms that do not use partial derivatives.

The idea of construction of explicit p^{th} *order Runge-Kutta methods* lies in obtaining approximate solution using the formula $v_{i+1} = v_{i+1} + h\varphi(x_i,u_i,h)$, where $\varphi(x_i,u_i,h)$ is a certain function approximating a Taylor series segment to the p^{th} order and not containing any partial derivatives of *f(x,u)*. The method is called *m-stage* based on the number of function value computations at a single step. This lecture gives formulas for the Runge-Kutta second-order and fourth-order methods. The most common among the family of Runge-Kutta methods is the four-stage Runge-Kutta fourth-order method.

Now let us discuss approaches to local truncation error reduction. One of approaches to step determination is as follows. Let the absolute local error for a p^{th} order method be within $\varepsilon > 0$. Then, *according to the Runge principle*, compute from x_i the values v_{i+1} at a step sized *h* and $\bar{v}_{i+1}$ at two steps sized *h/2*. After this, the step will vary depending on the value $S = \frac{\bar{v}_{i+1} - v_i}{2^p - 1}$ (it is doubled, halved or remains the same). Such an approach will enable control of the method local truncation error and vary the step *h* depending on the solution behaviour, however it is difficult enough as it requires multiple computations of the equation right-hand part.

The following issue covered by this lecture is another approach to ODE solving, i. e. the Adams method. At the i^{th} method step several approximations v_i of the solution $u(x)$ within a uniform mesh $x_i = x_0 + ih$ are considered to have been found, so the computation rule for the following approximation v_{i+1} is to be derived. For this purpose, the equation is integrated within the segment $[x_i, x_{i+1}]$, and $f(x,u(x))$ is interpolated by a polynomial of degree k. Depending on the set of points used for interpolation, i. e. the point x_j, $j = i–k,..., i$ or the point x_j, $j=i–k+1,..., I$ 1, the method is called either the extrapolation Adams-Bashforth method or the interpolation Adams-Moulton method. The local truncation error of Adams-Bashforth method is $O(h^{k+2})$ while its global truncation error is $O(h^{k+1})$. The *number of method steps* corresponds to the number of right-hand function values used in the computing formula. In case of the Adams-Bashforth methods, the number of method steps equals its order of accuracy; as for the Adams-Moulton methods, the number of steps is one less than the order of accuracy (except for the case when k= 0). The important difference between the extrapolation and interpolation Adams methods is that the extrapolation methods are explicit and the interpolation ones are implicit. This is why the interpolation Adams-Moulton methods can be used only in some particular cases, e. g. in case of u linearity of $f(x,u)$.

As a rule, both explicit and implicit methods (of the same or adjacent orders) are used together thus making *predictor-corrector* methods. These methods consist in prediction of the solution in the design point x_{i+1} using an explicit formula resulting in a rough approximation of the desired solution which is then corrected using an implicit formula with the predicted value in its right-hand part. As with the Adams method of any order only one new function value (or two if a predictor-corrector scheme is used) is to be computed to implement one step, the Adams methods of a relatively high order will be good for construction of the solution. However, this entails the problem of computing the first m–1 so called accelerating values $v_{1,...,} v_{m-1}$. To obtain them, the Runge-Kutta method with one step can be used.

Now we shall take a look at first-order methods of ODE system numerical solution that look like $U' = F(x,U)$, $U(x_0) = U^0$, where $U = (u_1, u_2, ..., u_n)$, $U' = (u_1', u_2', ..., u_n')$, $U^0 = (u_1^0, u_2^0, ..., u_n^0)$. Any of the methods described above is applicable to vector differential equations. In this case, in the formulas that define the methods only the independent variable x and step h are scalars; vectors sized n correspond to all other values. This lecture defines computing formulas for the Euler and Runge-Kutta methods to solve ODE systems.

Parallelization of methods to solve ODE systems is discussed further. Thus, the Euler method parallelization is generally possible only at the level of a single method iteration. If all the system equations are grouped into blocks whose number is equal to that of program threads, each thread will compute the solution components only within the respective block. The proposed scheme requires thread synchronization after one iteration, so it will be efficient only in case the right-hand part $F(x,U)$ is difficult to compute. Otherwise, most of the time will be dedicated to parallelism-related overheads and the program will demonstrate low efficiency.

In general, the Runge-Kutta method may be parallelized only within the single method step, not even within a single iteration. This also entails considerable overheads related to creation/closing of parallel sections after each iteration in case of fast right-hand part computation $F(x,U)$ and is only efficient when the right-hand part of the system is difficult to compute.

Then it will be proposed to study PDE solution by partial discretization. The method consists in transformation of a PDE into an ODE system by replacing partial derivatives by their approximation constructed using $v_i(t) \approx u(x_i, t)$, $1 \le i \le n–1$. To find these functions $v_1(t), v_2(t), ..., v_{n-1}(t)$ an ODE system will be used. This method can be illustrated by the heat equation. Simplification of the equation $\frac{\partial u}{\partial t} = \frac{\partial^2 u}{\partial x^2}$ will result in the ODE system $v'(t) = \frac{1}{h^2} Av(t)$

, where *A* is a tridiagonal matrix. The Euler method applied to the system corresponds to the complete discretization explicit difference scheme.

As the matrix A is sparse, arguments of the function f_i in the right-hand part of the equation will lie only in a certain neighbourhood of *i*. Then we shall introduce the notion of the *access distance d(F)* which is the smallest integer, for which any function $f_j(x,U)$ from the right-hand part can access only the subset $\{w_{j-b},..., w_{j+b}\}$ of the vector *U* components. If the access distance is restricted, i. e.*d(F)* << *n*, computations can be pipelined. The lecture demonstrates a pipelined computation scheme for the fourth-order Runge-Kutta method. In this case, if all system equations are grouped into n_B blocks whose number is equal to that of threads, computation of the block of functions $J = \{1,..., n_B\}$ requires the use of *J*–1, *J* and *J*+1 only (except for the case of adjacent blocks). For this purpose, only $n + O(n_B)$ stored values are enough.

RECOMMENDATIONS FOR STUDENTS

Study [**Ошибка! Источник ссылки не найден.** – 3] for detailed descriptions of the numerical methods of solving ODEs and ODE systems, method error analysis and method step size control procedures.

REFERENCES

1. Butcher, J.C. Numerical Methods for Ordinary Differential Equations. New York: John Wiley & Sons, 2003.
2. Hoffman, J.D. Numerical Methods for Engineers and Scientists, 2nd Edition. New York: CRC Press, 2001.
3. Kincaid D.R., Cheney E.W. Numerical Analysis: Mathematics of Scientific Computing, 3rd Edition. Pacific Grove: Brooks Cole, 2001.

PRACTICE

1. Implement the Euler method and the Runge-Kutta second-order and fourth-order methods. Implement the methods to solve a first-order test equation, compare the solutions and errors. Compare runtimes.
2. Implement the Adams-Bashforth and Adams-Moulton methods. Implement the methods to solve a first-order test equation, compare the solutions and errors. Compare the results to those of Task 1.
3. Implement the Euler method and the Runge-Kutta second-order and fourth-order methods to solve ODE systems. Use these methods to solve a first-order test system. Compare the runtime and truncation errors.
4. Parallelize the implemented ODE system numerical solution methods. Evaluate efficiency of the implementations.
5. Implement the partial discretization method to solve the heat transfer problem. Apply it to the test problem. Parallelize the program using a pipelined scheme. Evaluate efficiency of the parallel version.

TEST

1. What is the Euler method local truncation error order for the step size *h*?
    a. First
    b. + Second

   c. Third
2. What is the main disadvantage of the correcter Euler method?
   a. Low approximation order
   b. +Necessity to compute partial derivatives of *f(x,u)* at each step
   c. Necessity of multiple computations of the *f(x,u)* value at each step
3. The Runge-Kutta method is called *m-stage* if...
   a. It approximates ODE solution at the order *m*
   b. Its numerical solution requires a mesh of *m* points
   c. +To obtain v_i, the right-hand part function value is to be computed *m* times
4. How, according to Runge's rule, will the numerical method step *h* change if the auxiliary value *S* fulfills the condition of $2^{p+1} < |S| \le \varepsilon$?
   a. The step will double
   b. The step will be halved
   c. +The step will remain the same
5. What function is interpolated by the polynomial at the i^{th} step of the Adams methods?
   a. + $f(x,u(x))$
   b. $u(x)$
   c. $\int_{x_i}^{x_{i+1}} f(x,u(x))dx$
6. What is the local truncation error of the Adams-Bashforth method when a *k*-th order polynomial is used to interpolate functions and a meshsized *h*?
   a. + $O(h^{k+2})$
   b. $O(h^{k+1})$
   c. $O(h^k)$
7. What is the main idea of predictor-corrector methods?
   a. + An approximate problem solution is computed in the design point x_{i+1} using an explicit formula and corrected using an implicit formula with the predicted value in its right-hand part.
   b. An approximate problem solution is computed in the design point x_{i+1} using an implicit formula and corrected using an explicit formula with the predicted value in its right-hand part.
   c. An approximate problem solution is computed in the design point x_{i+1} using an explicit formula and corrected using the Adams-Bashforth method.
8. When the Euler or the second-order Runge-Kutta method can be efficiently parallelized to solve ODE systems by distribution of the vector *V* components among the threads?
   a. In case of a greater order of the system *n*
   b. In case of fast right-hand part computation $F(x,U)$
   c. +When the right-hand part of the system $F(x,U)$ is difficult to compute
9. Which difference scheme corresponds to the use of the implicit Euler method to solve an ODE system resulting from partial discretization of the heat equation?
   a. Explicit difference scheme
   b. +Implicit difference scheme
   c. Crank-Nicolson scheme
10. When does an ODE system allow pipelined parallelization?

a. If the system matrix *A* is a band matrix
b. +Any function $f_j(x,U)$ from the right-hand part has access only to the subset $\{w_{j-b},..., w_{j+b}\}$ of the vector *U* components and the access distance *d(F)<<n.*
c. If the system matrix *A* is sparse