Queens College, CUNY, Department of Computer Science

Numerical Methods

CSCI 361 / 761

Fall 2017

Instructor: Dr. Sateesh Mane

October 31, 2017

14 Lecture 14

Numerical solution of systems of ordinary differential equations

- We study numerical algorithms to solve systems of coupled ordinary differential equations.
- Most of the algorithms were motivated by problems of physics or engineering, although the algorithms themselves have wide applicability across many fields.
- In general, a differential equation does not have a unique solution. To specify a unique solution, we require some additional information, which is external to the differential equation.
- The problems to be solved can broadly be classified into two categories, depending on the nature of that external information.
- First, there are **initial value problems**.
  1. The typical scenario in this case is that the solution is specified at an initial time t_0 and the differential equation must be solved to obtain the solution at later times $t > t_0$.
  2. Many problems of physics fall into this category, such as the motion of the planets in the solar system.
  3. The independent variable in such problems is typically visualized as time, say t.
  4. The typical mental model in this situation is of "motion" from an initial starting time to later times.
  5. We frequently visualize the set of differential equations as **equations of motion.**
- Next, there are **boundary value problems**.
  1. The typical scenario in this case is that the value of the solution is specified at two points in space, say $x = a$ and $x = b$, which are the "end points" of the system, and the differential equation must be solved to obtain the solution at points in between $a < x < b$.
  2. Many problems of physics and engineering also fall into this category, for example the bending of a loaded beam clamped at two ends.
  3. The independent variable in such problems is typically visualized as space, say x.
  4. The typical mental model in this situation is of a "static" problem.
- This lecture is introductory, and will discuss general concepts pertaining to the numerical solution of systems of coupled ordinary differential equations.

14.1 First order differential equations and coupled systems of equations

- Almost all of the algorithms in the literature apply to **first order differential equations** and the equations are **coupled sets of differential equations.**
- The following example will explain why. Let the independent variable be x and consider a second order ordinary differential equation for $y(x)$ of the form

$$a_2(x,y)\,\frac{d^2y}{dx^2} + a_1(x,y)\,\frac{dy}{dx} = f(x,y)\,. \tag{14.1.1}$$

- Note that the equation is not necessarily linear.
- Introduce two variables $y_1(x)$ and $y_2(x)$, where $y_1 = y$ and rewrite eq. (14.1.1) in the form

$$\frac{dy_1}{dx} = y_2\,. \tag{14.1.2a}$$

$$a_2(x,y_1)\,\frac{dy_2}{dx} = -a_1(x,y_1)\,y_2 + f(x,y_1)\,. \tag{14.1.2b}$$

- Then eq. (14.1.2) is equivalent to eq. (14.1.2).
- However eq. (14.1.2) is a set of *two* coupled first order ordinary differential equations.
- For this reason, the algorithms in the literature describe the integration of coupled sets of first order ordinary differential equations.
- At least formally, we can divide the second equation by $a_2(x,y)$ to obtain coupled equations of the form (with an obvious notation for b and c)

$$\frac{dy_1}{dx} = y_2\,. \tag{14.1.3a}$$

$$\frac{dy_2}{dx} = b(x,y_1)\,y_2 + c(x,y_1)\,. \tag{14.1.3b}$$

- It should be clear from the above example that any ordinary differential equation, say of order n, can be reexpressed as a set of n coupled first order ordinary differential equations in variables $(y_1,\ldots,y_n)$, where $y_1 = y$ and

$$\frac{d}{dx}\begin{pmatrix} y_1 \\ y_2 \\ \vdots \\ y_n \end{pmatrix} = \begin{pmatrix} f_1(x,y_1,\ldots,y_{n-1}) \\ f_2(x,y_1,\ldots,y_{n-1}) \\ \vdots \\ f_n(x,y_1,\ldots,y_{n-1}) \end{pmatrix}\,. \tag{14.1.4}$$

- The additional variables $y_2,\ldots,y_n$ are called **auxiliary variables.**
- It should be obvious that the choice of auxiliary variables is not unique. If one set of auxiliary variables leads to difficulties in the numerical algorithm, try a different set.
- Typically, one has background information about the problem to be solved, which will suggest a guide to select a set of suitable auxiliary variables.

14.2 Existence of solutions Part 1

- Consider the following initial value problem. The differential equation is

$$\frac{dy}{dx} = \frac{1}{x}\,. \tag{14.2.1}$$

- The general solution is (here c is an arbitrary constant)

$$y = \ln(x) + c\,. \tag{14.2.2}$$

- Now suppose the initial condition is $y(0) = 0$.
- We cannot find a value for the constant c in eq. (14.2.2) which satisfies the initial condition $y(0) = 0$.
- Hence not every initial value problem has a solution.
- The initial conditions may be incompatible with the differential equation.
- There are mathematical theorems which specify conditions to guarantee the existence (and uniqueness) of a solution for an initial value problem.
- The matter will be discussed in more detail in future lectures.

14.3 Existence of solutions Part 2

- Consider the following boundary value problem. The differential equation is

$$\frac{d^2y}{dx^2} = -y\,. \tag{14.3.1}$$

- The general solution is (here a and b are arbitrary constants)

$$y = a\cos(x) + b\sin(x)\,. \tag{14.3.2}$$

- The equation is to be solved for $0 \le x \le 2\pi$.
- Suppose the boundary conditions are $y(0) = 0$ and $y(2\pi) = 1$.
- We cannot find values for the constants a and b in eq. (14.3.2) which satisfy both conditions $y(0) = 0$ and $y(2\pi) = 1$.
- To satisfy the condition $y(0) = 0$, we require $a = 0$. Then the solution is

$$y = b\sin(x)\,. \tag{14.3.3}$$

- However, at $x = 2\pi$, we find that $\sin(2\pi) = 0$, hence $y(2\pi) = 0$ for all values of b.
- Hence we cannot satisfy both the boundary conditions $y(0) = 0$ and $y(2\pi) = 1$.
- Conversely, if the boundary conditions $y(0) = 1$ and $y(2\pi) = 1$, then $a = 1$ and

$$y = \cos(x) + b\sin(x)\,. \tag{14.3.4}$$

- Now there are *infinitely many solutions*. Any value of b will satisfy the boundary conditions $y(0) = 1$ and $y(2\pi) = 1$.
- Hence if $y(0) = 1$ and $y(2\pi) = 1$, the solution is not unique and there are infinitely many solutions.
- Hence not every boundary value problem has a solution, or a unique solution.
- Boundary value problems can be more difficult to solve than initial value problems.
- As the above example shows, the general solution in eq. (14.3.2) is a smooth function. In fact it is an analytic function. The boundary conditions are not obviously unreasonable. Nevertheless, the boundary value problem has no solution.
- For this reason, it is more difficult to formulate mathematical theorems to guarantee the existence of solutions for boundary value problems.

14.4 Initial value problems: explicit & implicit methods

- Let $\boldsymbol{y}$ denote an array of n variables $(y_1, \ldots, y_n$ and let $y' = d\boldsymbol{y}/dx$.
- Consider the set of first order coupled ordinary differential equations

$$\boldsymbol{y}' = \boldsymbol{f}(x, \boldsymbol{y})\,. \tag{14.4.1}$$

- Let us integrate eq. (14.4.1) using a finite difference numerical derivative with a stepsize h.
- Let us employ a forward finite difference, from the step i to $i+1$, so

$$\boldsymbol{y}' \simeq \frac{\boldsymbol{y}_{i+1} - \boldsymbol{y}_i}{h}\,. \tag{14.4.2}$$

- Substituting into eq. (14.4.1) yields

$$\boldsymbol{y}_{i+1} = \boldsymbol{y}_i + h\,\boldsymbol{f}(x_i, \boldsymbol{y}_i)\,. \tag{14.4.3}$$

- This is an **explicit algorithm** or **explicit method.**
- All the terms on the right hand side of eq. (14.4.3) are known, hence the left hand $\boldsymbol{y}_{i+1}$ can be computed from known quantities.
- However, that is not the only possible choice. Suppose we employ a backward finite difference and write instead

$$\frac{\boldsymbol{y}_{i+1} - \boldsymbol{y}_i}{h} = \boldsymbol{f}(x_{i+1}, \boldsymbol{y}_{i+1})\,. \tag{14.4.4}$$

- Note that the terms on right hand side of eq. (14.4.4) are at the step $i+1$, not i.
- Rearranging terms yields

$$\boldsymbol{y}_{i+1} - h\,\boldsymbol{f}(x_{i+1}, \boldsymbol{y}_{i+1}) = \boldsymbol{y}_i\,. \tag{14.4.5}$$

- This is an **implicit algorithm** or **implicit method.**
- We must solve eq. (14.4.5) to find a solution for $\boldsymbol{y}_{i+1}$, for example via a root-finding algorithm such as multivariate Newton-Raphson.
- The root-finding algorithm will itself introduce a numerical error into the solution for $\boldsymbol{y}_{i+1}$, in addition to the finite-difference discretization error using the stepsize h.
- The use of an implicit algorithm seems peculiar, given that it requires extra computation at each step, and the result contains an additional numerical error beyond the finite-difference discretization error.
- However, implicit algorithms can have better stability and convergence properties than explicit algorithms, which is why they have survived and continue to be used by some experts, to solve practical problems.

14.5 Initial value problems: mixed explicit–implicit methods

- In general, a method is **fully explicit** if it contains no implicit calculations (this is obvious).
- Conversely, a method is **fully implicit** if it contains no explicit calculations.
- However, many implicit algorithms do contain some explicit calculations.
  They are not fully implicit.
- For example, let us average the explicit and implicit algorithms in eqs. (14.4.3) and (14.4.5) above. After all, we know that the centered finite difference has a higher order of accuracy that the forward and backward finite differences.
- This is called (not surprisingly) an **explicit-implicit method.**
- One can also say it is a **mixed explicit-implicit method.**
- The averaged equation is obtained by summing eqs. (14.4.3) and (14.4.5) and dividing by 2:

$$\boldsymbol{y}_{i+1} - \frac{h}{2}\,\boldsymbol{f}(x_{i+1}, \boldsymbol{y}_{i+1}) = \boldsymbol{y}_i + \frac{h}{2}\,\boldsymbol{f}(x_i, \boldsymbol{y}_i)\,. \tag{14.5.1}$$

- Because of the presence of $\boldsymbol{f}(x_{i+1}, \boldsymbol{y}_{i+1})$ on the left hand side, eq. (14.5.1) still requires the solution of an implicit equation.
- However, there are better ways to construct higher order integration algorithms.
- *Why stop there?* Define an interpolation parameter θ and form a weighted average. Then we obtain the equation

$$\boldsymbol{y}_{i+1} - \theta h\,\boldsymbol{f}(x_{i+1}, \boldsymbol{y}_{i+1}) = \boldsymbol{y}_i + (1-\theta)h\,\boldsymbol{f}(x_i, \boldsymbol{y}_i)\,. \tag{14.5.2}$$

- If $\theta = 0$ the method is fully explicit.
- If $\theta = 1$ the method is fully implicit.

14.6 Integrating forward and back

- Consider an initial value problem, and suppose the equations have been integrated from an initial value x_0 to a final value x_n after n steps.
- A good check on the accuracy of the solution is to **integrate the equations backwards to** x_0 and check if the solution matches the original initial values.
- In general, after integrating forwards and back, the numerical solution will *not* exactly equal the initial values.
- This will give an estimate of the numerical accuracy of the solution.

14.7 Boundary value problems: conditions on solution

- It was stated above that for a boundary value problem, the value of the solution is specified at two points in space, say $x = a$ and $x = b$, which are the "end points" of the system, and the differential equation must be solved to obtain the solution at points in between $a < x < b$.
- The above description is a bit simplistic.
- If the system consists of n coupled ordinary differential equations in the variables $(y_1, \ldots, y_n)$, then there must be totally n boundary conditions on the solution.
- However, there can be n_a boundary conditions at $x = a$ and n_2 boundary conditions at $x = b$, where $n_1 + n_2 = n$, and in general n_1 need not be equal to n_2.
    1. Obviously we must have $n_1 > 0$ and $n_2 > 0$, else the problem is an initial value problem.
    2. The boundary conditions can also be ***functions*** of x and $(y_1, \ldots, y_n)$.
    3. At $x = a$ we have a set of n_1 functions ("B" for "boundary")

$$B_{aj}(x = a, y_1(a), \ldots, y_n(a)) = 0\,. \qquad (j = 1, \ldots, n_1) \tag{14.7.1}$$

    4. At $x = b$ we have a set of n_2 functions

$$B_{bj}(x = b, y_1(b), \ldots, y_n(b)) = 0\,. \qquad (j = 1, \ldots, n_2) \tag{14.7.2}$$

- Note that the conditions B_{aj} and B_{bj} can be nonlinear equations.
- However, they will not be differential equations.
- It is also possible to specify conditions on the solution at intermediate points $x = c_1, c_2, \ldots$, where $a < c_1 < c_2 < \cdots < b$.
- They are technically not "boundary" conditions, but they express the idea that the conditions on the solution are specified at more than one value of x.

14.8 Boundary value problems: iterative methods

- The solution of boundary value problems typically requires iteration.
- For an initial value problem, we are given sufficient data (initial values) to begin the integration and propagate the solution forward.
- For a boundary value problem, we do not have enough initial data to do that.
- One strategy is to ***guess*** a set of initial values at $x = a$ and integrate to $x = b$ and check if the solution satisfies the boundary conditions at $x = b$.
    1. Almost certainly the boundary conditions at $x = b$ will not be satisfied.
    2. Hence we iterate our initial guess (using a root finding algorithm) until the boundary conditions at $x = b$ are satisfied to some tolerance.
    3. This is called a **shooting method.**
    4. We guess a set of initial values, and "shoot" the solution forward and see if it "hits" the target, which is the set of boundary conditions at $x = b$.
    5. The integration algorithm to propagate the solution from $x = a$ to $x = b$ can be an explicit method, implicit method, or mixed explicit–implicit method.
- Another strategy is to select a trial function which satisfies all the boundary conditions at both $x = a$ and $x = b$, and check if the function satisfies the system of differential equations.
    1. Almost certainly the function will not satisfy the system of differential equations.
    2. Hence we iterate the trial function until the system of differential equations is satisfied to some tolerance.
    3. For make this strategy work, we need to define a concept of "convergence" to say that the system of differential equations has been satisfied to a sufficient approximation over an interval $a \leq x \leq b$.
    4. This is called a **relaxation method.**
    5. We guess an initial trial function, and iterate until it "relaxes" to the solution of the system of differential equations.
    6. Note that by construction, the initial trial function and all the iterates always satisfy all the boundary conditions. It is the system of differential equations which is not satisfied.
- Hence these are the two basic strategies:
    1. In a shooting method, the system of differential equations is satisfied, and we iterate until the boundary conditions are satisfied, up to some tolerance.
    2. In a relaxation method, the boundary conditions are satisfied, and we iterate until the system of differential equations is satisfied, up to some tolerance.
- The definition of "convergence" and "tolerance" is different in the two strategies.

14.9 High accuracy and high order

- Let the stepsize be h, when integrating to solve the differential equations.
- **We say that an integration method is of order k if the numerical error in the solution, in one integration step, is of order $O(h^{k+1})$.**
- Let $\boldsymbol{y}_{\text{exact}}(x)$ be the exact solution of the system. Then for a method of order k,

$$\boldsymbol{y}_{i+1} - \boldsymbol{y}_i = \boldsymbol{y}_{\text{exact}}(x_{i+1}) - \boldsymbol{y}_{\text{exact}}(x_i) + O(h^{k+1})\,. \tag{14.9.1}$$

- For example, for a first order method (such as eqs. (14.4.3) or (14.4.5)), the numerical error in one integration step is $O(h^2)$.
- Many widely used methods are of *fourth* order.
  1. The numerical error in one integration step is $O(h^5)$.
  2. Fourth order methods yield a substantial improvement over first order methods, with only a modest increase in computational complexity.
  3. We shall study some fourth order methods in these lectures.
- Note that the above definition of "order" neglects roundoff error. The numerical error of "$O(h^{k+1})$" is *only due to the finite stepsize h.*
- For an implicit method, the above definition of "order" also neglects the numerical error caused by the approximate solution of the implicit equations.
- **An important lesson to bear in mind is this:**

  Higher order does not always imply higher accuracy.
- The derivation of a high order integration algorithm usually assumes that the underlying functions are smooth and differentiable to a high order. This may not always be true.
- For example, consider a system of second order differential equations. A fourth order integration method may require that the functions have continuous *fourth derivatives,* and this may not be the case. A sixth order integration method may require that the functions have continuous *sixth derivatives,* and this may be false.

14.10 Variable stepsize

- The stepsize h does not have to be the same for every integration step.
- The value of the stepsize h_i can depend on i.
- **Variable stepsize methods** are widely used.
- For an integration algorithm of order k, the numerical error of a variable stepsize methods at the step i is $O(h_i^{k+1})$.
- Because h_i depends on i, it can be difficult to calculate a global error analysis for a variable stepsize method, to determine the global error in the solution summing over all the integration steps.
- An example is the motion of a planet sround the Sun (or a satellite around the Earth). On the second orbit, the step size used by the integration algorithm may not be the same as on the first orbit, and similarly for the other orbits.
- Performing a global error analysis for a variable stepsize method is more difficult than for a fixed stepsize method.

14.11 Conservation laws, symmetries and invariants

- In many problems of physics, there are fundamental laws such as conservation of energy.
- This means the value of the total energy is an invariant, whose value does not change as a particle moves.
- Hence when the equations of motion for a particle are integrated, the value of the energy remains equal to its initial value and does not change with time.
- Conservation laws mean there are invariants associated with the system.
- In general, there may be multiple invariants associated with a given set of differential equations.
- In the exact solution of the differential equations, the values of the invariants do not change and remain equal to their initial values.
- In a numerical integration procedure, the differential equations are integrated only approximately, hence in general the values of the invariants will not be preserved exactly.
- A general numerical integration procedure does not know anything about invariants, but if we know that some invariants exist for the set of differential equations, such knowledge can be employed to design a better integration algorithm.
- It is also possible that there are some mathematical symmetries associated with the system of differential equations, or with the solution of the equations. Knowledge of such symmetries can also be employed to design a better integration algorithm.
- It sometimes happens that a system has two invariants but it is not possible to design an integration algorithm which preserves the values of both invariants together. We may have to choose one invariant or the other. There are multiple examples in physics of this nature, and rigorous mathematical theorems on the subject.