

12.010 Computational Methods of Scientific Programming Lecture 11: Homework 2 solution

Lecturers
Thomas A Herring
Chris Hill

Discussion of HW 02 solution

- Today's lecture we will run through aspects of the solutions to Homework 2.
- Three problems:
 - Table of Legendre polynomials and associated functions
 - Reading and format of character strings
 - Modeling the trajectory of a paper plane
- Today we run through the solutions with an opportunity to ask questions about the methods used.

Problem 1: Legendre functions

- Method:
 - Explicit coding of functions
 - Recursion relationships: Multiple to choose from. Solution has two difference ones implemented.
- Other features
 - Format generation into string
 - Small offsets from -1 and +1 because second recursion algorithm does not work at these limits (Test this in class)

10/20/2005

12.010 Lec 11

3

Problem 2: Names

- Relatively simple problem. Issues related to:
 - How to convert to upper and lower case
 - What to do with user input
 - Two styles of input used.
 - How to output just the pieces of the strings that we want.
- Can the program be broken?

10/20/2005

12.010 Lec 11

4

Problem 3: Paper plane

- Correction to force equations from Homework solution
 1. Problem still solvable but not realistic
 - Added Wing Area explicitly (originally was to be absorbed into lift and drag coefficients but this was not done)
- Integration methods: Runge-Kutta methods
- Selection of step size using the difference between two results with different step sizes. We can experiment with this
- Testing and evaluating program.

10/20/2005

12.010 Lec 11

5

Runge-Kutta integration

- Compare the two versions of the second-order system
- $y'' = f(x, y, y')$

$$y_{n+1} = y_n + h[y'_n + \frac{1}{6}(k_1 + k_2 + k_3)] + O(h^5)$$

$$y'_{n+1} = y'_n + \frac{1}{6}[k_1 + 2k_2 + 2k_3 + k_4]$$

$$k_1 = hf(x_n, y_n, y'_n)$$

$$k_2 = hf(x_n + h/2, y_n + hy'_n/2 + hk_1/8, y'_n + k_1/2)$$

$$k_3 = hf(x_n + h/2, y_n + hy'_n/2 + hk_1/8, y'_n + k_2/2)$$

$$k_4 = hf(x_n + h, y_n + hy'_n + hk_3/2, y'_n + k_3)$$

10/20/2005

12.010 Lec 11

6

Form with no Velocity dependence

$$\bullet y'' = f(x,y)$$

$$y_{n+1} = y_n + h[y'_n + \frac{1}{6}(k_1 + 2k_2)] + O(h^4)$$

$$y'_{n+1} = y'_n + \frac{1}{6}k_1 + \frac{2}{3}k_2 + \frac{1}{6}k_3]$$

$$k_1 = hf(x_n, y_n)$$

$$k_2 = hf(x_n + h/2, y_n + hy'_n/2 + hk_1/8)$$

$$k_3 = hf(x_n + h, y_n + hy'_n + hk_2/2)$$

10/20/2005

12.010 Lec 11

7

Integration continued

- Notice the way the problem is formulated.
- h is the step size (i.e., the time step the integrator is going to use)
- In the first form of Runge-Kutta, the error in the y_{n+1} is $O(h^5)$
- This means the error is "of the order of" h to the 5th power.
- If we halve the step size h , then the error should be approximately 32 times smaller.
- (See the `num_int.f` routine where a similar relationship was used but in that case we could quantify the error)
- In the second form of Runge-Kutta, the error was $O(h^4)$ meaning that halving the step size would reduce the error by approximately 16 times.
- How are we going to use this? (Think about what was done in `num_int.f`).

10/20/2005

12.010 Lec 11

8

Integration error

- In some cases, the actual size of the error in the numerical integration can be determined based on the high order derivative of the function (as was done in num_int.f and is done in the Matlab quad function)
- In many cases, these formulas are very complex and can take a long time to compute (relative to the numerical integration itself).
- You can obtain an approximate estimate of the error in the integration by looking at the difference between the results using two step sizes.
- The easiest implementation is to use h and $h/2$ as the two step sizes.
- Using h , the integrator is step once to time $t+h$ and using $h/2$ two steps are taken to $t+h$
- If the difference in y between h and $h/2$ is δy , then the error in the integrator with step size h is $\Delta y = \delta y / (1 - 1/2^n)$ where n is 4 or 5 depending on the integrator used.

10/20/2005

12.010 Lec 11

9

Integrator error continued

- Using this difference between the results with two step sizes, we can estimate the error in one step of the integrator
- For long integrations, it is important to note the error is not random but tends to retain the same sign with each step and so will grow as the system is integrated.
- If the errors were random, their accumulated effects would grow as the square-root of the number of steps.
- A better approximation is the accumulated error will grow linearly with time.
- An estimate of the error is bounded between these values
- Using this approach, we can monitor the error.
- (Sometimes this approach is used to correct the integrator, thus increasing its performance by another power of h --- but the error in corrected integrator is difficult to assess.
- What next in our design of the solution?

10/20/2005

12.010 Lec 11

10

Other aspects to consider

- If we will possibly have a changing step size to keep the integration errors low, then what do we do about output of the results (or passing the results to an internal graphics routine)?
- The “result output” rate (i.e., the time interval for output the results) will differ from the integration step size most likely. How will be specify this?
- Options:
 - Have the user specify time interval for output
 - Have the user specify a distance that must travel before the results are output?
- Which is better? Pros and cons to each approach. If time given than maybe too many points are generated that can never really be displayed. If distance, then time step in not the same and an animation of the results will give the wrong impression of the speed the objects are moving with.

10/20/2005

12.010 Lec 11

11

Evaluation

- The rest of the class will be spent examining the homework solution and playing with programs.
- Student results will also be examined for those willing to have their programs shown and tested.

10/20/2005

12.010 Lec 11

12