Numerical Methods

ODEs – II

10.05.2016

ODE's (cont'd): Numerical soln methods for IVP-ODE: (Initial Value Problem)

Key Concepts: - Distinguish explicit techniques from implicit ones.

- Local truncation error $\Leftrightarrow$ order
- Stability of the numerical method.

IVP-ODE: Many physical applications lead to higher order ODEs; they can be reformulated to 1st order ODE systems

$$\underline{y}'(t) = \frac{d\underline{y}(t)}{dt} = \underline{f}(t, \underline{y})$$

$$\underline{y}(t_0) = \underline{y}_0$$

makes the solution is unique.

General ODE: $y(t)$: unknown; nth order ODE:

$$a\frac{d^n y(t)}{dt^n} + b\frac{d^{n-1} y(t)}{dt^{n-1}} + \cdots + \frac{dy(t)}{dt} = f(t, y, y', \ldots)$$

$a(y)$

* There is only 1 independent variable in an ODE. (eg. t)

Difference from a PDE, which has multiple independent variables.

eg.

Predator / Prey:

prey: $\frac{du}{dt} = \alpha u - \gamma v u$ — −ve interaction

predator: $\frac{dv}{dt} = -\beta v + \sigma u v$ — +ve interaction.

Homework — Solve in

add initial values of $u(t=0)$, $v(t=0)$

Define the coeff. $\alpha, \beta, \gamma, \sigma > 0$.

Numerically solve this system of ODEs.

Summary:

Forward Euler Method:

$$y' = f(t, y(t)), \quad y(t_0) = y_0$$

Explicit

$$y_{i+1} = y_i + h f(t_i, y_i)$$

$$y_0 = y(t_0)$$

$$t_{i+1} = t_i + h \Rightarrow \text{step size}$$

$+ \frac{h^2}{2} f''(\xi) \Rightarrow$ Method $O(h)$

$t_0 \to t_0 + h \to t_0 + 2h \to t_0 + 3h$

$y(\ ) \to y(\ ) \to y(\ )$

Backward Euler Method: $O(h)$

Implicit

$$y_{i+1} = y_i + h f(t_{i+1}, y_{i+1})$$

$$y_0 = y(t_0)$$

$$t_{i+1} = t_i + h \Rightarrow \text{step size}$$

Absolute Stability Criterion: $y' = \lambda y$, $\lambda < 0 \Rightarrow$

$|y_{i+1}| \leq |y_i| \Rightarrow$ Forward-Euler.

$$h \leq \frac{2}{|\lambda|}$$

but Backward-Euler method: $h > 0$

Ex: Solve numerically using backward Euler.

$$\left.\begin{aligned} y' &= y^2 - t \\ y(0) &= -1 \end{aligned}\right\} \quad h = 0.5$$

Let

$$y_{i+1} = y_i + h f(t_{i+1}, y_{i+1})$$

$$(\star)\quad \begin{cases} y_{i+1} = y_i + h(y_{i+1}^2 - t_{i+1}) \\ t_{i+1} = t_i + h \\ y_0 = -1 \end{cases}$$

$$y_{i+1} - h y_{i+1}^2 = \boxed{y_i - h t_{i+1}} \triangleq a \quad : \text{RHS given}$$

Let $x = y_{i+1} \Rightarrow$ $\boxed{F(x) - a = 0}$ ← solve this by a root finding technique

e.g. use Newton (-Rhapson) method or fixed-point method to solve for $y_{i+1} = x$

Let $g(x) = h x^2 + a = x$.

$$x_{k+1} = g(x_k) \Rightarrow$$

$$x_1 = h x_0^2 + a$$

$$x_2 = h x_1^2 + a$$

$$\vdots$$

$$x_k = \cdots \rightarrow \text{converges}$$

Result y_{i+1} ✓

Continue in this fashion, i.e. in each iteration of $(\star)$, solve a possibly nonlinear equation:

$$y_{i+1} - y_i - h f(t_{i+1}, y_{i+1}) = 0$$

to get y_{i+1}.

Runge-Kutta (RK) Methods : Higher Order :

$y' = f(t, y) \Rightarrow$ Take Integral from t_i to t_{i+1}

(★) $y(t_{i+1}) = y(t_i) + \int_{t_i}^{t_{i+1}} f(t, y(t))\, dt$

discretize the integral by basic quadrature methods.

eg. use

Implicit Trapezoidal rule :

$$y(t_{i+1}) = y(t_i) + \frac{h}{2}\left[f(t_i, y(t_i)) + f(t_{i+1}, y(t_{i+1}))\right]$$

(Implicit)

Local truncation error : $O(h^2)$:

2nd order RK methods : Let's derive 2 methods :

(1) Use forward Euler to approximate $f(t_{i+1}, y(t_{i+1}))$

first approximate $y(t_{i+1})$ in $f(\cdot,\cdot)$ $\approx \boxed{Y = y_i + h \cdot f(t_i, y_i)}$: 1st step. $O(h)$

Explicit Trapezoidal rule $y_{i+1} = y_i + \frac{h}{2}\left[f(t_i, y_i) + f(t_{i+1}, Y)\right]$

(2) Apply midpoint quadrature rule to (★)

$\Rightarrow y_{i+1} = y_i + h\, f\left(t_{i+\frac{1}{2}}, y_{i+\frac{1}{2}}\right)$

Use forward Euler to approximate $y_{i+\frac{1}{2}} = y_i + \frac{h}{2} f(t_i, y_i)$

$\Longrightarrow$

Explicit RK method: midpoint

$$y = y_i + \frac{h}{2} f(t_i, y_i)$$

(2) $$y_{i+1} = y_i + h f(t_{i+\frac{1}{2}}, y)$$

y_0 is given

* At each step, we first evaluate y, then y_{i+1}: 2 function evaluations/step $\therefore$ roughly twice as expensive as the forward Euler method/step.

but $O(h^2)$ vs $O(h)$

Classical 4th order RK: (4 stage) 4 fn. evaluations/step.

$O(h^4)$: Derivation based on Simpson quadrature rule (not derive here).

$$y_1 = y_i$$
$$y_2 = y_i + \frac{h}{2} f(t_i, y_1)$$
$$y_3 = y_i + \frac{h}{2} f(t_{i+\frac{1}{2}}, y_2)$$
$$y_4 = y_i + h f(t_{i+\frac{1}{2}}, y_3)$$
$$y_{i+1} = y_i + \frac{h}{6}\left[f(t_i, y_1) + 2f(t_{i+\frac{1}{2}}, y_2) + 2f(t_{i+\frac{1}{2}}, y_3) + f(t_{i+1}, y_4)\right]$$

Given an initial condn y_0.

Explicit RK methods: (they are conditionally stable).

BLG 202E Numerical Methods

Learning Objectives: Students will be able to:

1. Define errors in numerical analysis; define discretization error and compare to round-off errors in computing; Construct floating point representation and relate to precision; connect it to round-off errors;
2. Roughly identify computational complexity of algorithms via O(.) notation
3. Define condition number of a matrix and describe its utility in assessing conditioning of a given problem such as solution of a linear system of equations
4. Construct an iterative solution to find the zeros of a nonlinear equation through techniques such as Fixed Point Iteration, Newton's method, Secant method; describe its connection to optimization; analyze convergence of iterative methods
5. Construct algorithms for solution of linear system of equations through both Gaussian elimination, forward/backward substitution, and LU decomposition; define their complexities; describe the advantage of LU decomposition in solution of linear systems
6. Formulate a linear least squares problem to solve an overdetermined system of equations; derive its solution via normal equations ; apply to data fitting problems
7. Construct the spectral decomposition (Eigen Value Decomposition-EVD) and Singular Value Decomposition (SVD) of matrices, and truncated SVD. Describe and use Power method to estimate the dominant eigen value/vector of a given matrix; Apply eigen (singular) value /vector and truncated SVD methods to real life problems;
8. Describe principles of data interpolation and approximation; Expand the interpolating/approximating function over a set of basis functions; construct polynomial interpolation solution using different polynomial bases such as monomial, Lagrange, Newton polynomials; apply those to real problems

Learning Objectives (continued):

9. Derive formulae for approximation of numerical derivatives at different orders using Taylor series method and Richardson extrapolation method; Similarly formulate numerical differentiation using polynomial interpolation method; Analyze and discuss effects of round-off errors and data errors (noise) in numerical differentiation

10. Formulate numerical integration using both basic and composite quadrature algorithms: trapezoidal rule, midpoint rule, and Simpson's rule, and how those are obtained using polynomial interpolation

11. Construct numerical solutions to initial value problems in systems of ordinary differential equations (ODEs). Apply the explicit methods such as forward Euler method, and implicit methods such as backward Euler method to solve those ODEs, and describe the stability criteria for those methods. Apply higher order methods, specifically the Runge Kutta 2nd order and 4th order methods, to solution of initial value ODE problems.

12. Implement all concepts in Learning Objectives 1-11 in a programming environment (such as MATLAB)