微分方程数值解课程项目报告
项目 1: 刚性方程数值方法对比

胡维佳

22339032

2025 年 4 月 14 日

1 问题描述

考虑刚性微分方程初值问题：

$$y' = -1000y + 3000 - 2000e^{-t}, \quad y(0) = 0 \tag{1}$$

其真实解（下文会验证）为：

$$y(t) = 3 - \frac{2000}{999}e^{-t} - \frac{997}{999}e^{-1000t} \tag{2}$$

任务要求为：1. 设定不同时间步长，绘制数值解 2. 分析三种方法的绝对稳定区域 3. 用真实解（请验证其为真实解）计算全局误差：

$$\|E(h)\|_\infty = \max |y_{数值} - y_{真实}| \tag{3}$$

4. 分析误差随步长变化的行为，数值验证方法的收敛阶

2 数值方法

本项目采用以下三种数值方法进行求解：

2.1 显式欧拉方法

$$y_{n+1} = y_n + hf(t_n, y_n) \tag{4}$$

2.2 隐式欧拉方法

$$y_{n+1} = y_n + hf(t_{n+1}, y_{n+1}) \tag{5}$$

2.3 改进欧拉方法

$$y_{n+1} = y_n + \frac{h}{2}[f(t_n, y_n) + f(t_{n+1}, y^*_{n+1})] \tag{6}$$

其中 $y^*_{n+1} = y_n + hf(t_n, y_n)$ 为预测值。

3 稳定性分析

由课本 P40 定义 5.1：如果

$$\rho(\lambda) - \overline{h}\sigma(\lambda) = 0 \tag{7}$$

的根都在单位圆内，则称线性 k 步法关于 $\overline{h}$ 绝对稳定。若存在复数域 D_A，使多步法对 $\forall\overline{h} \in D_A$ 都绝对稳定，则称 D_A 为绝对稳定域。

3.1 显式欧拉方法

3.1.1 离散格式

$$y_{n+1} = y_n + hf(t_n, y_n) = y_n + h\lambda y_n = (1 + h\lambda)y_n \tag{8}$$

3.1.2 稳定性条件

- 要求放大因子 $|1+h\lambda| \leq 1$。
- 设 $z = h\lambda$，则稳定区域为：$|1+z| \leq 1$。
- 在复平面上，这是以 $(-1, 0)$ 为中心、半径为 1 的圆内区域。

3.1.3 刚性方程分析

- 对于刚性方程，参数 $\lambda = -1000$。
- 显式欧拉法的稳定性条件为：

$$|1 - 1000h| \leq 1 \quad \Rightarrow \quad h \leq 0.002$$

- 结论：步长 h 必须极小（≤ 0.002），表明显式方法对刚性方程不适用。

3.2 隐式欧拉方法

3.2.1 离散格式

$$y_{n+1} = y_n + hf(t_{n+1}, y_{n+1}) = y_n + h\lambda y_{n+1} \tag{9}$$

解得：

$$y_{n+1} = \frac{y_n}{1 - h\lambda} \tag{10}$$

3.2.2 稳定性条件

- 要求放大因子 $|\frac{1}{1-h\lambda}| \leq 1$。
- 设 $z = h\lambda$，则稳定区域为：$|1-z| \geq 1$。
- 在复平面上，这是以 $(1, 0)$ 为中心、半径为 1 的圆外区域。

3.2.3 刚性方程分析

当 $\mathrm{Re}(\lambda) < 0$ 时，对所有 $h > 0$ 均满足稳定性条件，因此隐式欧拉是无条件稳定的，特别适合求解刚性方程。

3.3 改进欧拉方法

3.3.1 离散格式

$$y_{n+1} = y_n + \frac{h}{2}[f(t_n, y_n) + f(t_{n+1}, y^*_{n+1})] = y_n + \frac{h}{2}(\lambda y_n + \lambda y_{n+1}) \tag{11}$$

其中 $y^*_{n+1} = y_n + hf(t_n, y_n)$ 为预测值。解得：

$$y_{n+1} = \frac{1 + \frac{h\lambda}{2}}{1 - \frac{h\lambda}{2}} y_n \tag{12}$$

3.3.2 稳定性条件

- 要求放大因子 $|\frac{1+\frac{h\lambda}{2}}{1-\frac{h\lambda}{2}}| \leq 1$。
- 设 $z = h\lambda$，则稳定区域为：$\frac{1+\frac{z}{2}}{1-\frac{z}{2}}| \leq 1$。
- 等价于 $|1 + \frac{z}{2}| \leq |1 - \frac{z}{2}|$。
- 在复平面上，这表示所有实部小于 0 的点 (左半平面)。

3.3.3 刚性方程分析

- 理论上，改进欧拉法和隐式欧拉法的绝对稳定区域相同（均为左半平面）。
- 实际数值实验中，由于显式预测步的存在，改进欧拉法对刚性方程（如 $\lambda \ll 0$）仍需满足 $h \leq \frac{2}{|\lambda|}$ 才能避免振荡。

表 1: 三种数值方法的稳定性比较

方法	绝对稳定区域	适合刚性方程
显式欧拉	$\lvert 1+h\lambda \rvert \leq 1$	不适合
隐式欧拉	$\mathrm{Re}(h\lambda) < 0$	非常适合
改进欧拉	$\mathrm{Re}(h\lambda) < 0$（理论上）	条件适合

3.4 三种数值方法稳定性对比

4 数值实验与结果

4.1 不同时间步长下的数值解（要求 1）

- 三种数值方法的 matlab 函数：
  - 显式欧拉方法函数

Listing 1: MATLAB 示例代码

```
function [t, y] = expliciteuler(f, tspan
    , y0, h)
%显式欧拉法求解ODE: dy/dt = f(t, y)
t_start = tspan(1);
t_end = tspan(2);
t = t_start:h:t_end;
y = zeros(size(t));
y(1) = y0;
%迭代
for n = 1:length(t)-1
y(n+1) = y(n) + h * f(t(n), y(n));
end
end
```

– 隐式欧拉方法函数

Listing 2: MATLAB 示例代码

```
function [t, y] = impliciteuler(f, tspan
    , y0, h, tol, maxiter)
if nargin<5
tol = 1e-6;
end
if nargin<6
maxiter = 1000;
end
%隐式欧拉法求解ODE: dy/dt = f(t, y)
t_start = tspan(1);
t_end = tspan(2);
t = t_start:h:t_end;
y = zeros(size(t));
y(1) = y0;
%迭代
for n = 1:length(t)-1
t_new = t(n+1);
y_prev = y(n);
%牛顿迭代求解y_{n+1}
y_next = y_prev;%初始猜测
for k = 1:maxiter
F = y_next-y_prev-h*f(t_new, y_next);%定
    义隐式方程
dfdy = -1000;%线性方程自己计算
dFdy = 1-h*dfdy;
delta = -F/dFdy;
y_next = y_next+delta;
%检查收敛
if abs(delta)<tol
```

```
        break;
    end
end
y(n+1) = y_next;
end
end
```

– 改进欧拉方法函数

Listing 3: MATLAB 示例代码

```
function [t, y] = improvedeuler(f, tspan, y0, h)
%改进欧拉法求解ODE: dy/dt = f(t, y)
t_start = tspan(1);
t_end = tspan(2);
t = t_start:h:t_end;
y = zeros(size(t));
y(1) = y0;
%迭代
for n = 1:length(t)-1
y_p = y(n)+h*f(t(n), y(n));%预测步
y(n+1) = y(n)+(h/2)*(f(t(n), y(n))+f(t(n+1), y_p));%校正步（梯形）
end
end
```

- 主程序：引用上述函数

Listing 4: MATLAB 示例代码

```
f = @(t, y)-1000*y+3000-2000*exp(-t);
y_exact = @(t)3-(2000/999)*exp(-t)-(997/999)*exp(-1000*t);
```

```matlab
%要求1.设定不同时间步长，绘制数值解
tspan = [0, 0.1];
y0 = 0;
h_values = [0.001, 0.002, 0.005, 0.01, 0.02,
    0.1];
figure;
for i = 1:length(h_values)
h = h_values(i);

[t1, y1] = expliciteuler(f, tspan, y0, h);
[t2, y2] = impliciteuler(f, tspan, y0, h);
[t3, y3] = improvedeuler(f, tspan, y0, h);

%绘制数值解
subplot(3, 2, i);
plot(t1, y1, 'r-', t2, y2, 'g--', t3, y3, 'b-',
    t1, y_exact(t1), 'k-');
title(['h=', num2str(h)]);
legend('显式欧拉', '隐式欧拉', '改进欧拉', '真实
    解');
xlabel('t');
ylabel('y(t)');
grid on;
end
```

- 结果图像对比（图 1）

4.2 全局误差分析

4.2.1 验证真实解（用代码）

Listing 5: MATLAB 示例代码

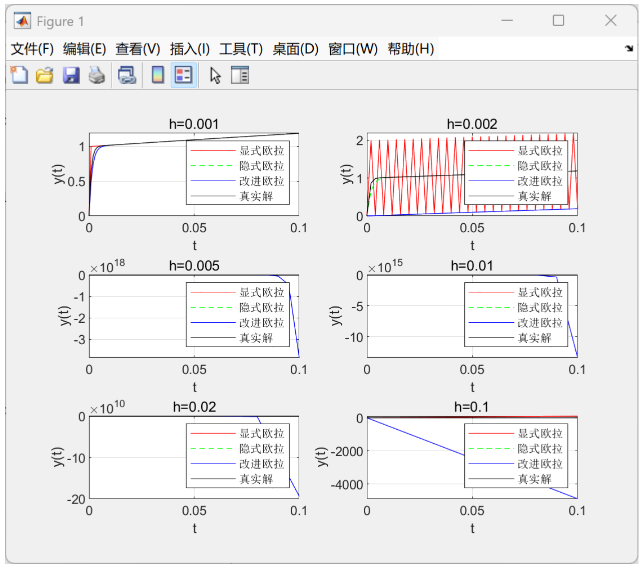

图 1: 不同数值方法在不同步长下的解对比

```
syms t;
% 定义解析解
y_exact = 3 - (2000/999)*exp(-t) - (997/999)*exp(-1000*t);

% 计算解析解的导数
dy_exact = diff(y_exact, t);

% 定义微分方程的右边
f = -1000*y_exact + 3000 - 2000*exp(-t);

% 检查 dy_exact 是否等于 f
simplify(dy_exact - f)
```

运行结果为 0，是真实解，验证完毕。

4.2.2 计算全局误差

- 代码

Listing 6: MATLAB 示例代码

```
f = @(t, y)-1000*y+3000-2000*exp(-t);
y_exact = @(t)3-(2000/999)*exp(-t)-(997/999)
    *exp(-1000*t);
tspan = [0, 0.1];
y0 = 0;
%要求3.计算全局误差
h_values = logspace(-6, -3, 20);
errors = zeros(3, length(h_values));
for i = 1:length(h_values)
h = h_values(i);
try
[t1, y1] = expliciteuler(f, tspan, y0, h);
[t2, y2] = impliciteuler(f, tspan, y0, h);
[t3, y3] = improvedeuler(f, tspan, y0, h);

errors(1, i) = max(abs(y1-y_exact(t1)));
errors(2, i) = max(abs(y2-y_exact(t2)));
errors(3, i) = max(abs(y3-y_exact(t3)));
catch ME
warning('Error at h=%g: %s', h, ME.message)
errors(:,i) = NaN;
end
end
% 显示误差对比表
valid_idx = ~any(isnan(errors));
```

```matlab
        T = table(h_values(valid_idx)', errors(1, valid_idx)', errors(2, valid_idx)', errors(3, valid_idx)', ...
        'VariableNames', {'StepSize', 'ExplicitEuler', 'ImplicitEuler', 'ImprovedEuler'});
        disp('Global Errors (||E(h)||_inf):');
        disp(T);

        % 导出表格为图片
        tableText = evalc('disp(T)');
        f_table = figure('Position', [100 100 600 300], 'Color', 'white');
        axis off;
        text(0, 1, tableText, 'FontName', 'Monospaced', 'FontSize', 10, ...
        'VerticalAlignment', 'top', 'Interpreter', 'none');
        exportgraphics(f_table, 'GlobalErrors_Table.png', 'Resolution', 300);
        close(f_table);

        % 绘制误差曲线
        figure;
        loglog(h_values(valid_idx), errors(1, valid_idx), 'ro-', 'DisplayName', 'Explicit Euler'); hold on;
        loglog(h_values(valid_idx), errors(2, valid_idx), 'bs-', 'DisplayName', 'Implicit Euler');
        loglog(h_values(valid_idx), errors(3, valid_idx), 'gd-', 'DisplayName', '
```

```
                                Improved Euler');
                            xlabel('Step Size (h)');
                            ylabel('Global Error ||E(h)||_\infty');
                            legend('Location', 'best');
                            title('Error vs. Step Size for Stiff ODE');
                            grid on;
```

- 误差分析表

StepSize	ExplicitEuler	ImplicitEuler	ImprovedEuler
1e-06	0.00018365	0.0001835	6.1237e-08
1.4384e-06	0.00026422	0.0002639	1.2675e-07
2.0691e-06	0.00038016	0.00037951	2.6239e-07
2.9764e-06	0.00054705	0.0005457	5.4328e-07
4.2813e-06	0.00078734	0.00078453	1.1252e-06
6.1585e-06	0.0011334	0.0011276	2.3315e-06
8.8587e-06	0.0016322	0.0016202	4.8341e-06
1.2743e-05	0.0023517	0.0023269	1.0031e-05
1.833e-05	0.0033908	0.0033394	2.0844e-05
2.6367e-05	0.004894	0.0047877	4.3392e-05
3.7927e-05	0.0070747	0.0068539	9.0564e-05
5.4556e-05	0.01025	0.0097917	0.00018976
7.8476e-05	0.014888	0.013955	0.00039986

图 2: 误差分析表

- 误差随步长变化图像（见图 3）

4.3 收敛阶分析

- 观察图 3，有全局误差 $E = O(h^p)$
- 收敛阶数通常与全局误差阶数 p 一致

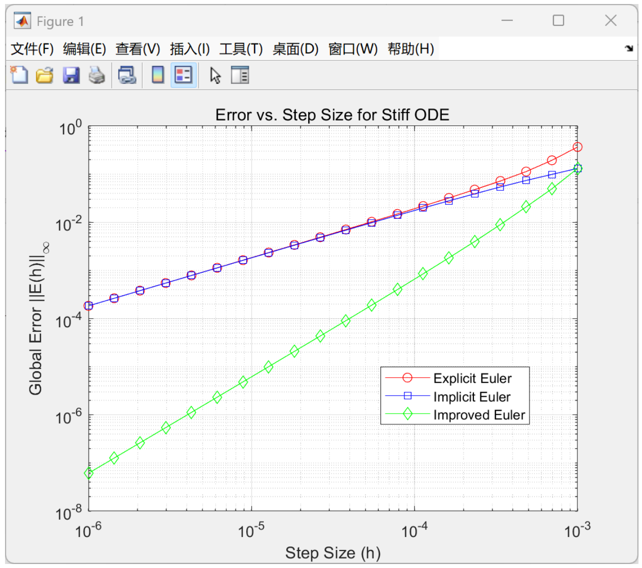

图 3: 误差随步长变化图像

- 由此可见，显式欧拉法的收敛阶为 1 阶，隐式欧拉法的收敛阶为 1 阶，改进欧拉法的收敛阶为 2 阶。