# Homework 2016-03-18

Chuan Lu
13300180056

March 21, 2016

**Problem 1.**
Analysis the stability and absolute stable interval of Modified and Revised Euler Iteration for Dahlquist test.

*Proof.* **0.1 For Modified Euler Iteration:**

$$u_{n+1} = u_n + \frac{\Delta t}{2}(f_n + f_{n+1})$$

Assume that the disturbed initiate value is $\widetilde{u_0}$ and the exact initiate value is $u_0$. Mark the following value calculated by the Modified Euler Iteration as $\widetilde{u_i}, i = 1, 2, \cdots$. Then $\widetilde{u_n} = \widetilde{u_{n-1}} + \frac{\Delta t}{2}(a\widetilde{u_{n-1}} + a\widetilde{u_n})$, $\quad \widetilde{u_n} = \frac{1+\frac{a\Delta t}{2}}{1-\frac{a\Delta t}{2}}\widetilde{u_{n-1}}$, $\quad u_n = \frac{1+\frac{a\Delta t}{2}}{1-\frac{a\Delta t}{2}}u_{n-1}$. Hence, $\|\widetilde{u_n} - u_n\| = \frac{1+\frac{a\Delta t}{2}}{1-\frac{a\Delta t}{2}}\|\widetilde{u_{n-1}} - u_{n-1}\| = \cdots = (\frac{1+\frac{a\Delta t}{2}}{1-\frac{a\Delta t}{2}})^n\|\widetilde{u_0} - u_0\|$. If we expect that to a fixed $\Delta t$, when $n \to \infty$, the error can still be of control, then it must be true:

$$\|\frac{1 + \frac{a\Delta t}{2}}{1 - \frac{a\Delta t}{2}}\| \leqslant 1.$$

Hence, because that $\Delta t \geqslant 0$, it can only be true when $a \leqslant 0$, and $0 \leqslant \Delta t < \frac{2}{|a|}$.
On the other hand, take $z = a\Delta t$, we have $\|\frac{2+z}{2-z}\| \leqslant 1, \quad z \in \mathcal{Z}$.

## 0.2 For Revised Euler Iteration:

$$u_{n+1} = u_n + \Delta ta(u_n + \frac{\Delta t}{2}au_n)$$

Claimed as above, we have $\widetilde{u_{n+1}} = \widetilde{u_n}(1+\Delta ta+\frac{1}{2}(\Delta ta)^2)$. Hence, $\|\widetilde{u_n} - u_n\| = (1+\Delta ta+\frac{1}{2}(\Delta ta)^2)^n\|\widetilde{u_0} - u_0\|$. As shown above, $\|1+\Delta ta+\frac{1}{2}(\Delta ta)^2\| \leqslant 1$. On the other hand, take $z = a\Delta t$, we have $\|z^2+2z+2\| \leqslant 2$. ☐

**Problem 2.**
Prove that Implicit Euler Iteration is of first-order convergence for Dahlquist test.

*Proof.* According to (2.2.20), $u_n = \frac{1}{(1-a\Delta t)^n}u_0$. For $e_n = u(t_n) - u_n$, we have $e_n = u_0(e^{at_n} - e^{-nln(1-a\Delta t)})$. From Taylor expansion we can know that

$$e_n = -u_0e^{at_n}(\frac{a^2t_n}{2}\Delta t + O(\Delta t^2)).$$

In consequence, for a fixed $T = N\Delta t$, $\|u(T) - u(N)\| \leqslant C\Delta t$, which means the implicit iteration is of first-order convergence. ☐

**Problem 3.**
Calculate $\frac{du}{dt} = au$ with Euler Iterations, and show the convergence by pics.

*Proof.* **0.3 The code is shown as follows.**

```
function [ t, u ] = Euler_iter( func, inteval, u0, delta_t, op )
% EULER_ITER The main function of Euler Iteration of solving ODEs
% The equation behave likes du/dt = f(t, u), with initial condition given
% as u(0) = u0 in the inteval [a, b];

% input:
% func : a function of two variables t, u;
% inteval : a list of the inteval of the equation, given like [a, b];
% u0 : the initial condition;
% delta_t : the step size of time;
% op : the kind of iterations, chosen from "explicit', 'implicit',
% 'modified', 'revised';

% output:
% t : the list of time, inited by the inteval and delta_t;
% u : the value of u at the points in t;

if nargin < 2
    error('More arguments needed --Euler-iter');
elseif nargin == 2
    u0 = 1;
    delta_t = 1/8;
    op = 'explicit';
elseif nargin == 3
    delta_t = 1/8;
    op = 'explicit';
elseif nargin == 4
    op = 'explicit';
end

if length(inteval) ~= 2 || inteval(1) >= inteval(2)
    error('Invalid inteval --Euler-iter');
end

switch op
    case 'explicit'
        [t, u] = explicit_iter(func, inteval, u0, delta_t);
    case 'implicit'
        [t, u] = implicit_iter(func, inteval, u0, delta_t);
    case 'modified'
        [t, u] = modified_iter(func, inteval, u0, delta_t);
    case 'revised'
        [t, u] = revised_iter(func, inteval, u0, delta_t);
    otherwise
        error('Invalid operation --Euler-iter');
end

end
```

```
function [ t, u ] = explicit_iter( func, inteval, u1, delta_t )
% EXPLICIT_ITER Explicit Euler Iteration

```

```
t = inteval(1):delta_t:inteval(2);
n = length(t);
u = zeros(1, n);

for i = 1 : n
    u(i) = u1;
    u1 = u1 + delta_t * feval(func, t(i), u(i));
end
```

```
function [t, u] = implicit_iter(func, inteval, u1, delta_t)
% IMPLICIT_ITER Implicit Euler Iteration

t = inteval(1):delta_t:inteval(2);
n = length(t);
u = zeros(1, n);
f = @(u, u0, t, delta_t)(u - delta_t * feval(func, t, u) - u0);

for i = 1 : n - 1
    u(i) = u1;
    t_i = t(i + 1);
    u0 = u1;
    u1 = fsolve(@(u) f(u, u0, t_i, delta_t), 1);
end
u(n) = u1;

end
```

```
function [t, u] = modified_iter(func, inteval, u1, delta_t)
% MODIFIED_ITER Modified Euler Iteration

t = inteval(1):delta_t:inteval(2);
n = length(t);
u = zeros(1, n);
f = @(u, u0, t, delta_t)(u - delta_t * feval(func, t, u) - u0);

for i = 1 : n - 1
    u(i) = u1;
    t_i = t(i + 1);
    u0 = u1;
    u1 = fsolve(@(u) f(u, u0, t_i, delta_t), 1);
    u1 = u0 + delta_t/2 * (feval(func, t(i), u0) + ...
        feval(func, t_i, u1));
end
u(n) = u1;

end
```

```
function [t, u] = revised_iter(func, inteval, u1, delta_t)
% REVISED_ITER Revised Euler Iteration

t = inteval(1):delta_t:inteval(2);
n = length(t);
u = zeros(1, n);

```

```
for i = 1 : n - 1
    u(i) = u1;
    u_half = delta_t * feval(func, t(i), u(i)) / 2 + u1;
    t_half = (t(i) + t(i + 1))/2;
    u1 = u1 + delta_t * feval(func, t_half, u_half);
end
u(n) = u1;

end
```

```
% Homework 4-3
clear all;
format long;
a = -2;
func = @(t, u)(a .* u);
u0 = 1;
u1 = -1;
delta_t = 1/8;
inteval = [0, 3];
op = {'explicit', 'implicit', 'modified', 'revised'};
symbol = {'*-', '.-', 'd-', 'o-'};

figure(1);
for i = 1:4
    [t, u] = Euler_iter(func, inteval, u0, delta_t, char(op(i)));
    [t2, u2] = Euler_iter(func, inteval, u1, delta_t, char(op(i)));
    plot(t, u, cell2mat(symbol(i)));
    hold on;
    plot(t2, u2, cell2mat(symbol(i)));
    hold on;
end

exact_func = @(x, u0)(exp(a .* x + log(u0)));
exact_value1 = feval(exact_func, t, u0);
exact_value2 = feval(exact_func, t, u1);
plot(t, exact_value1, 'x-');
hold on;
plot(t, exact_value2, 'x-');
legend('Explicit1', 'Explicit2', 'Implicit1', 'Implicit2',...
    'Modified1', 'Modified2', 'Revised1', 'Revised2',...
    'Exact1', 'Exact2', 'Location', 'Best');
title('Solving ODE du/dt = a*u with Euler Iteration');
grid on;

figure(2);
m = 5;
error_list = zeros(4, m);
for j = 1:m
    delta_t = 2 ^ (-j - 2);
    t = inteval(1):delta_t:inteval(2);
    exact_value = feval(exact_func, t, u0);
    for i = 1:4
        [t, u] = Euler_iter(func, inteval, u0, delta_t, char(op(i)));
        error_list(i, j) = max(abs(u - exact_value));
```

```
        end
end
for i = 1 : 4
        loglog(2 .^ (-3:-1:-m-2), error_list(i, :), cell2mat(symbol(i)));
        hold on;
end
legend('Explicit', 'Implicit', ...
        'Modified', 'Revised', ...
        'Location', 'Best');
title('Error of Euler iterations of solving du/dt = a*u');
grid on;
```

### 0.4 The result is shown as follows.

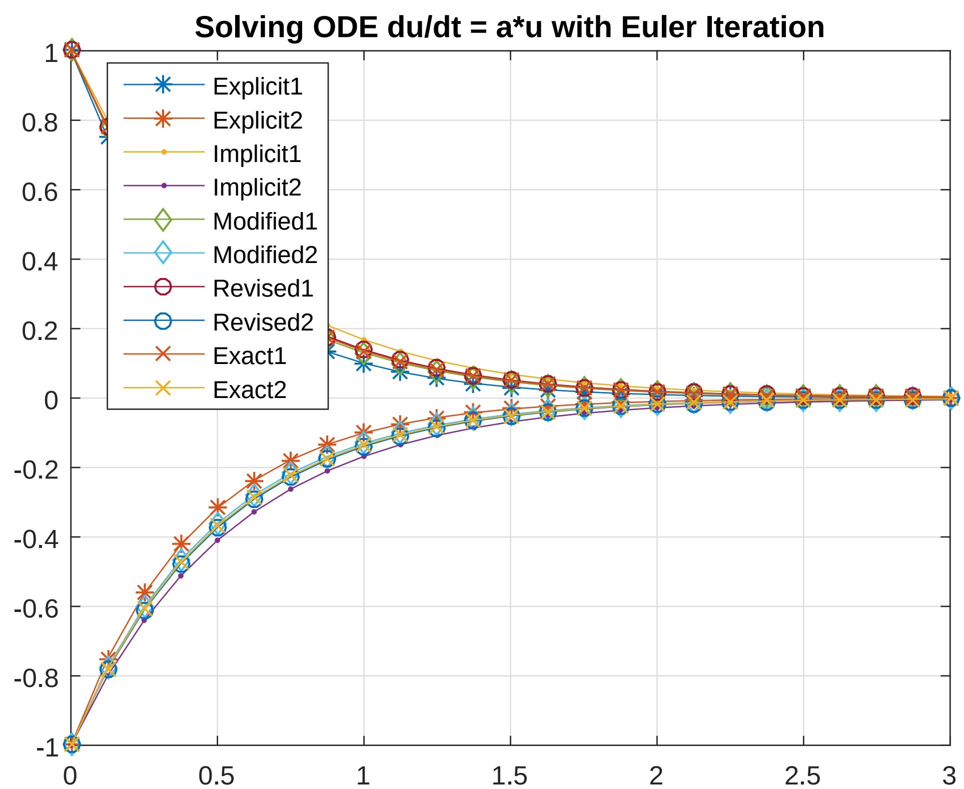


Figure 1: The convergence of Euler Iteration.

□

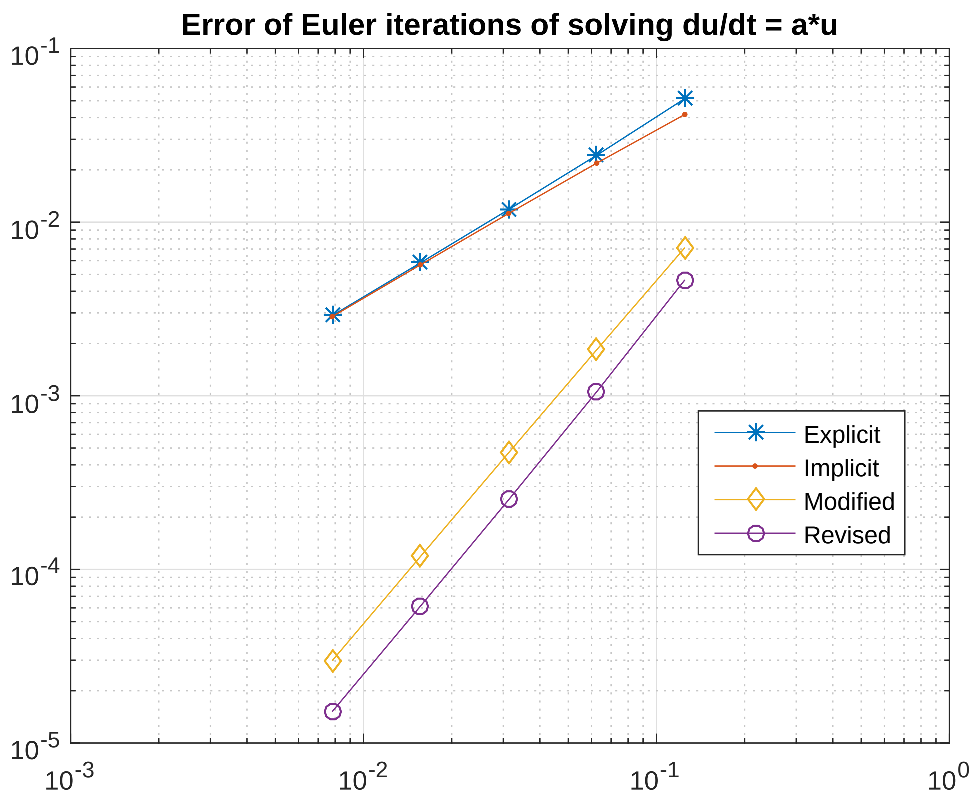


Figure 2: The error of Euler Iteration.