MATLAB数值实验：函数逼近法求方程的数值解

作者：凯鲁嘎吉 - 博客园 http://www.cnblogs.com/kailugaji/

这篇博客主要通过给定的数学迭代公式，利用MATLAB来迭代求解多项分数阶微分方程的数值解，主要用到的是函数逼近法，一种是非线性化数值解法，一种为线性化数值解法，并绘制解析解与数值解的函数图像，计算两者的误差。

1．问题描述

数值实验：函数逼近法求方程的数值解

➢ 问题描述(求u(t)) 求解多项分数阶微分方程的数值解法

$$ {}_0^C D_t^{\alpha_1} u(t) + {}_0^C D_t^{\alpha_2} u(t) + {}_0^C D_t^{\alpha_3} u(t) = f(t,u(t)) \qquad 0 \le t \le T $$

$$ u(0) = 0 $$

➢ 解析解

$$ u(t) = t^{2+\alpha_1} $$

➢ 数值解-公共表示 $\alpha_1 = 0.9,\ \alpha_2 = 0.6,\ \alpha_3 = 0.3,\ T = 2,\ l = 0.1$(步长)

$$ b_k^{(\alpha)} = \frac{1}{\Gamma(2-\alpha)}\left[(k+1)^{1-\alpha} - k^{1-\alpha}\right] \qquad k = 0,1,\cdots n-1 $$

$$ C = b_0^{(\alpha_1)} + l^{\alpha_1-\alpha_2} b_0^{(\alpha_2)} + l^{\alpha_1-\alpha_3} b_0^{(\alpha_3)} \qquad u(t_n) \to u_n $$

$$ f(t,u(t)) = \frac{\Gamma(3+\alpha_1)}{\Gamma(3)} t^2 + \frac{\Gamma(3+\alpha_1)}{\Gamma(3+\alpha_1-\alpha_2)} t^{2+\alpha_1-\alpha_2} + \frac{\Gamma(3+\alpha_1)}{\Gamma(3+\alpha_1-\alpha_3)} t^{2+\alpha_1-\alpha_3} + \sin(t^{2+\alpha_1}) - \sin(u) $$

1

➢ 数值解一(非线性)

$$u_n = u_{n-1} - \frac{1}{C}\sum_{k=0}^{n-2}\left(b_{n-k-1}^{(\alpha_1)} + l^{\alpha_1-\alpha_2} b_{n-k-1}^{(\alpha_2)} + l^{\alpha_1-\alpha_3} b_{n-k-1}^{(\alpha_3)}\right)[u_{k+1} - u_k] + \frac{1}{C} l^{\alpha_1} f(t_n, u_n)$$

$$1 \le n \le N$$

➢ 数值解二(线性)

$$u_1 = u_0 + \frac{1}{C} l^{\alpha_1} f(t_1, u_1)$$

$$u_n = u_{n-1} - \frac{1}{C}\sum_{k=0}^{n-2}\left(b_{n-k-1}^{(\alpha_1)} + l^{\alpha_1-\alpha_2} b_{n-k-1}^{(\alpha_2)} + l^{\alpha_1-\alpha_3} b_{n-k-1}^{(\alpha_3)}\right)(u_{k+1} - u_k) + \frac{1}{C} l^{\alpha_1} f(t_n, 2u_{n-1} - u_{n-2})$$

$$2 \le n \le N$$

公式来源：多项分数阶常微分方程的数值微分法
http://max.book118.com/file_down/e37129207cb5d8d84fa03b346b308819.docx

2

2. MATLAB程序

demo_1.m

```
clear
clc
format long % 数据形式为长精度
% Author: 凯鲁嘎吉 - 博客园 http://www.cnblogs.com/kailugaji/
%% 定义变量
alpha1 = 0.9;
```

```
alpha2 = 0.6;
alpha3 = 0.3; % 1>alpha1>alpha2>alpha3>0

%% 求解开始
T = 2; % 区间右端点
tau = 0.1; % 步长
TT = 0:tau:T; % t变量序列，也就是方程中的自变量t
N = length(TT)-1; % t变量序列个数-1

% 定义三个1*N的0矩阵用来储存方程中每一项的系数
b_alpha1 = zeros(1,N);
b_alpha2 = zeros(1,N);
b_alpha3 = zeros(1,N);

% 循环开始
for k = 0 : (N-1)
    b_alpha1(k+1) = ((1+k)^(1-alpha1)-(k)^(1-alpha1))/gamma(2-alpha1);
    b_alpha2(k+1) = ((1+k)^(1-alpha2)-(k)^(1-alpha2))/gamma(2-alpha2)*tau^(alpha1-alpha2);
    b_alpha3(k+1) = ((1+k)^(1-alpha3)-(k)^(1-alpha3))/gamma(2-alpha3)*tau^(alpha1-alpha3);
end

coe_0 = b_alpha1(0+1) + b_alpha2(0+1) + b_alpha3(0+1);

U = zeros(1,N+1); % 储存计算的结果
for n = 1:N
    temp = 0;
    for k = 0 : n-2
        temp = temp + (b_alpha1(n-k-1+1) + b_alpha2(n-k-1+1) + b_alpha3(n-k-1+1))*(U(k+1+1)-U(k+1));
    end
    temp0 = U(n);
    while 1
        temp1 = U(n-1+1) - temp /coe_0+ tau^(alpha1)*right_fun(TT(n+1),temp0,alpha1,alpha2,alpha3)/coe_0;
        % 计算误差 如果前一次迭代和后一次迭代的误差小于10^-7，那么久退出循环，并把最后一次迭代的值赋给U
        if abs(temp0-temp1) < 10^(-7)
            U(n+1) = temp1;
            break;
        else
            temp0 = temp1;
        end
    end
end
```

```
True_sol = true_fun(TT,alpha1); % 真实值
plot(TT,U,'-')
hold on
plot(TT,True_sol,'r*')
legend('数值解','解析解','Location','northwest')
title('Algorithm 1');
xlabel('t');
ylabel('u(t)');
err = max(abs(U-True_sol)); % 误差
saveas(gcf,sprintf('Algorithm 1.jpg'),'bmp'); %保存图片
fprintf('方法一中解析解与数值解之间的误差为：%f\n', err);

function aa = true_fun(t,alpha1)
aa = t.^(2+alpha1);
end
function bb = right_fun(t,u,alpha1,alpha2,alpha3)
bb = gamma(3+alpha1)/gamma(3)*t.^2+gamma(3+alpha1)/gamma(3+alpha1-alpha2)*t.^(2+alpha1-alpha2)+gamma(3+alpha1)/gamma(3+alpha1-alpha3)*t.^(2+alpha1-alpha3)+sin(t.^(2+alpha1))-sin(u);
end
```

demo_2.m

```
clear
clc
format long
% Author: 凯鲁嘎吉 - 博客园 http://www.cnblogs.com/kailugaji/
%% 定义变量
alpha1 = 0.9;
alpha2  = 0.6;
alpha3 = 0.3;   % 1 > alpha1 > alpha2 > alpha3 > 0
%% 求解开始
T = 2;
tau = 0.1;
TT = 0:tau:T;
N = length(TT)-1;
% 定义三个1*N的0矩阵用来储存方程中每一项的系数
b_alpha1 = zeros(1,N);
b_alpha2 = zeros(1,N);
b_alpha3 = zeros(1,N);
% 循环开始
for k = 0 :(N-1)
    b_alpha1(k+1) = ((1+k)^(1-alpha1)-(k)^(1-alpha1))/gamma(2-alpha1);
    b_alpha2(k+1) = ((1+k)^(1-alpha2)-(k)^(1-alpha2))/gamma(2-alpha2)*tau^(alpha1-alpha2);
```

```
    b_alpha3(k+1) = ((1+k)^(1-alpha3)-(k)^(1-alpha3))/gamma(2-alpha3)*tau^(alpha1-alpha3);
end
coe_0 = b_alpha1(0+1) + b_alpha2(0+1) + b_alpha3(0+1);
%%%%%%%%%%%%%%%%%%%%%%%%
U = zeros(1,N+1);
temp0=U(2);
%% 第一个值特殊处理
while 1
    temp1= tau^(alpha1)*right_fun(TT(1+1),temp0,alpha1,alpha2,alpha3)/coe_0 ;
    if (abs(temp0-temp1) < 10^(-7) )
        U(2) = temp1;
        break;
    else
        temp0 = temp1;
    end
end
%% 2~N个值的计算
for n = 2:N
    temp = 0;
    for k = 0 : n-2
        temp = temp + (b_alpha1(n-k-1+1) + b_alpha2(n-k-1+1) + b_alpha3(n-k-1+1))*(U(k+1+1)-U(k+1));
    end
    temp0 = U(n);
    while 1
        XX=2*U(n-1+1)-U(n-2+1);
        temp1 = U(n-1+1) - temp /coe_0+ tau^(alpha1)*right_fun(TT(n+1),XX,alpha1,alpha2,alpha3)/coe_0;
        % 计算误差 如果前一次迭代和后一次迭代的误差小于10^-7，那么久退出循环，并把最后一次迭代的值赋给U
        if (abs(temp0-temp1) < 10^(-7) )
            U(n+1) = temp1;
            break;
        else
            temp0 = temp1;
        end
    end
end
True_sol = true_fun(TT,alpha1); % 真实值
plot(TT,U,'-')
hold on
plot(TT,True_sol,'r*')
legend('数值解','解析解','Location','northwest')
title('Algorithm 2');
xlabel('t');
```

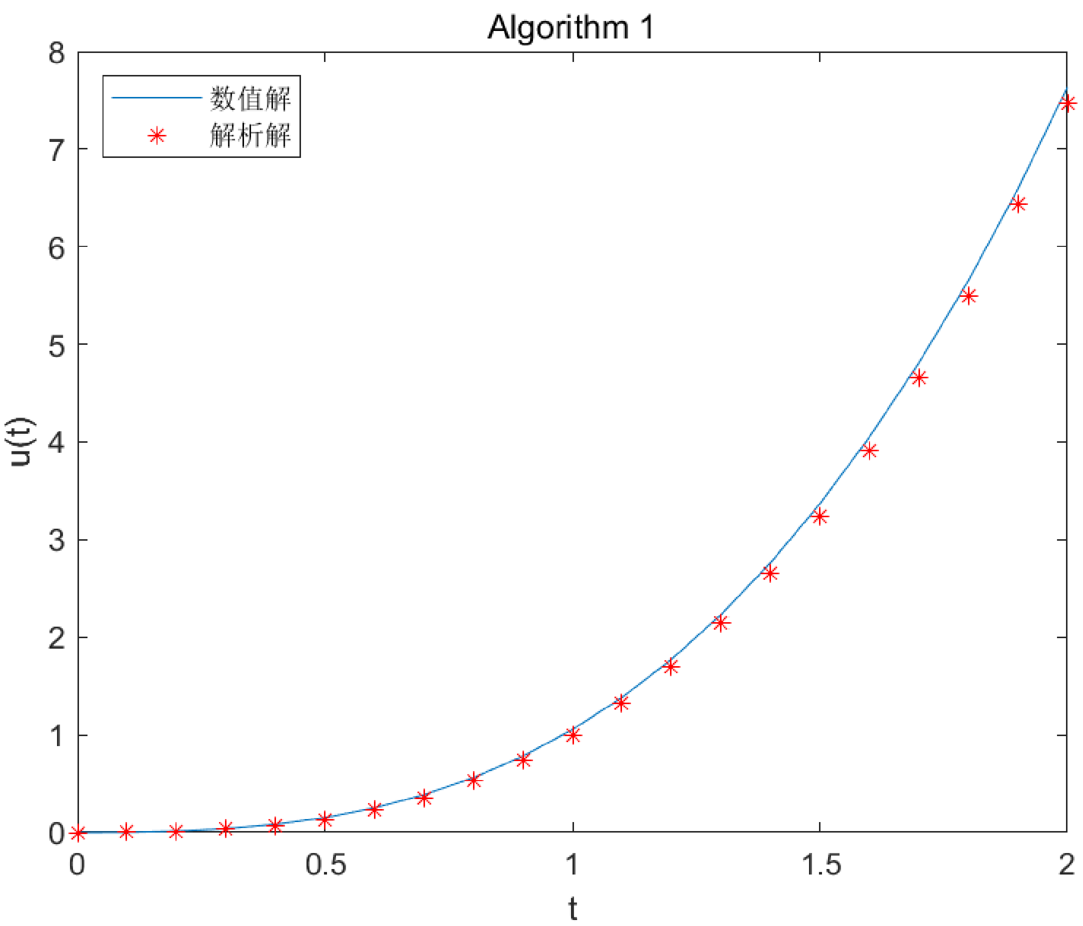
```
ylabel('u(t)');
err = max(abs(U-True_sol)); % 误差
saveas(gcf,sprintf('Algorithm 2.jpg'),'bmp'); %保存图片
fprintf('方法二中解析解与数值解之间的误差为：%f\n', err);

function aa = true_fun(t,alpha1)
aa = t.^(2+alpha1);
end
function bb = right_fun(t,u,alpha1,alpha2,alpha3)
bb = gamma(3+alpha1)/gamma(3)*t.^2+gamma(3+alpha1)/gamma(3+alpha1-alpha2)*t.^(2+alpha1-alpha2)+gamma(3+alpha1)/gamma(3+alpha1-alpha3)*t.^(2+alpha1-alpha3)+sin(t.^(2+alpha1))-sin(u);
end
```

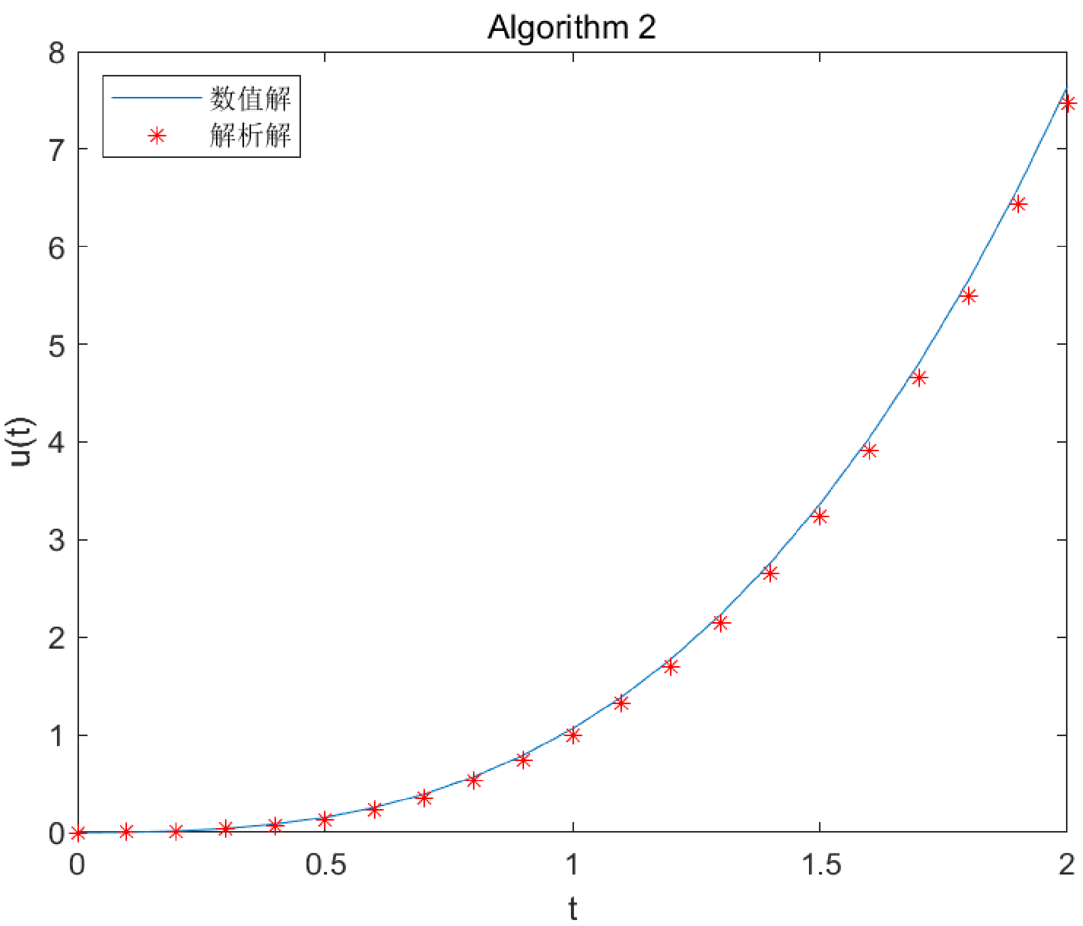
3．结果

```
>> demo_1
方法一中解析解与数值解之间的误差为：0.169468
>> demo_2
方法二中解析解与数值解之间的误差为：0.175177
```

方法一结果图



方法二结果图：



本博文问题来源：多项分数阶常微分方程的数值微分法 http://max.book118.com/file_down/e37129207cb5d8d84fa03b346b308819.docx