数学试验复习

向飞宇

2019 年 5 月 30 日

1 插值与拟合问题

插值问题使用 interp/spline/csape 等求解. 使用 fnplt，fnval，fnder 分别对函数绘图，求值，求导.

拟合问题使用 lsqcurvefit/lsqnonlin 求解. 此处给出使用 lsqcurvefit 的代码模板.

```
x = [## 自变量数据集 ##];
y = [## 观测数据集 ##];
f = @(c, x) ## c是待求变量，f是拟合函数 ##;
% 以下是优化设置，可以调整参数
options = optimset();
options.MaxFunEvals = 10000;
options.MaxIter = 10000;
options.TolX = 1e-10;
[c, resnorm, residual, exitflag, output] = lsqcurvefit(f, ##
↪ c的初值 ##, x, y, [], [], options);
c
```

2 常微分方程的求解

求解使用 ode45，代码框架如下.

```
options = odeset('RelTol',1e-12, 'AbsTol', [##
↪  根据解向量的维度确定 ##]); %
[t, y] = ode45(@(t, y)rigid(t, y), [## 求解区间 ##], [##
↪  解的初值 ##], options);

% 此处可以使用plot等函数画图
% 例如: plot(t, y)

function dy = rigid(t, y)
    dy = zeros(n, 1); % 初始化
    % 以下给出微分方程
    dy(1) = ..;
    dy(2) = ..;
    ...
    dy(n) = ..;
end
```

3 数值积分问题

3.1 Gauss 积分

以下是 Gauss 积分模板（仅限一元函数），此模板较为复杂，需要根据问题自己修改.

```
function result = generic_gauss_integral(funct, lb, ub,
↪  density, coeff, points, ilb, iub)
    % generic_gauss_integral calculates generic integrals
↪  using
    % Gauss-Legendre integral method.
    % funct: the function to be integrated
    % lb: lower bound of the integral, default = -1
    % ub: upper bound of the integral, default = 1
    % density: intervals of the integral segmentations,
    ↪  default = 1000
```

```
% coeff: coefficients of the approximation in the
↪  integral, default =
% Gauss-Legendre coefficients of 3-order precision
% points: nodes to use in the integral, default =
↪  Gauss-Legendre nodes
% of 3-order precision
% ilb: the lower bound of the approximation formula used,
↪  default = -1
% iub: the lower bound of the approximation formula used,
↪  default = 1

% Parameter completing
if (nargin == 1)
    lb = -1;
    ub = 1;
    density = 50;
    coeff = [1, 1];
    points = [sqrt(3) / 3, -sqrt(3) / 3];
    ilb = -1;
    iub = 1;
elseif (nargin == 3)
    density = 50;
    coeff = [1, 1];
    points = [sqrt(3) / 3, -sqrt(3) / 3];
    ilb = -1;
    iub = 1;
elseif (nargin == 4)
    coeff = [1, 1];
    points = [sqrt(3) / 3, -sqrt(3) / 3];
    ilb = -1;
    iub = 1;
elseif (nargin == 6)
    ilb = -1;
```

```
        iub = 1;
    end

    % Parameter checking
    if (lb > ub) || (density <= 0) || (size(coeff, 2) ~=
    ↪   size(points, 2))
        error('Incompatible parameters');
    end

    interval = (ub - lb) * density;
    result = 0;
    for i = 1: interval
        lb_now = lb + (i - 1) / interval * (ub - lb);
        ub_now = lb_now + (ub - lb) / interval;
        [coeff_a, coeff_b] = interval_transform(lb_now,
        ↪   ub_now, ilb, iub);
        result = result + interval_sum(@(x)funct(x), coeff,
        ↪   (points - coeff_b * ones(size(points))) /
        ↪   coeff_a) / coeff_a;
    end
end

function [coeff_a, coeff_b] = interval_transform(lb, ub, nlb,
↪   nub)
    coeff_a = (nub - nlb) / (ub - lb);
    coeff_b = -(nub * lb - nlb * ub) / (ub - lb);
end

function interval = interval_sum(funct, coeff, points)
    interval = 0;
    for i = 1: size(coeff, 2)
        interval = interval + coeff(i) * funct(points(i));
    end
```

```
end
```

注：带权函数的 Gauss 积分不能使用等分区间增加精度，即只有使用 Gauss-Legendre 多项式时才能使用等分区间. 无法增加精度时，将 density 的值设置为 1.

3.2 Monte-Carlo 积分

以下是 Monte-Carlo 积分模板（可以使用高维函数，但需用带分量的匿名函数写出[1]），使用均值法进行计算.

```
function value = monte_carlo(funct, dimension, iteration, lb,
↪   ub, constraints)
    if nargin == 5
        constraints = {};
    end
    % monte-carlo main method
    points = zeros(dimension, iteration);
    total_value = 0;
    for i = 1: dimension
        points(i, :) = generate_random_points(iteration,
        ↪   lb(i), ub(i));
    end
    % tag for constraints
    for j = 1: iteration
        tag = true;
        for k = 1: size(constraints, 1) % check constraints
            if (constraints{k}(points(:, j)') > 0)
                tag = false;
                break;
            end
        end
        if (tag)
            total_value = total_value + funct(points(:, j)');
```

[1] 例如：f = @(x)x(1) .* x(2) .* x(3)

```
        end
    end
    value = total_value * prod(ub - lb) / iteration;
end

function point = generate_random_points(numbers, lb, ub)
    % random point generator by uniform distribution
    point = lb + (ub - lb) .* rand(1, numbers);
end
```

注意，传入积分区域含有多个条件时，需要使用一个列级 cell 来表示.

4 函数图像的绘制

对于有特殊需要的图像绘制，需要使用相应函数，如 pcolor, contour, quiver 等等. 如果没有这种需求，plot，fplot，plot3 这样的函数足以满足需求.

以下是绘制比较清晰的图像的模板.

```
fplot/plot/plot3(.., 'linewidth', 1~1.5); % 画线
plot/plot3(.., 'x/o/.', 'MarkerSize', 8~10); % 画点
xlabel(..);
ylabel(..);
zlabel(..);
legend(['..'; '..'; '..'; ..]); % 图例长度需对齐，或使用 "",
↪  "构造字符串
set(gca, 'fontsize', 16); % 设置字号大小为16
```

此外，使用 PDF 格式或 EPS 格式对于提高清晰度很有帮助.

5 优化问题

优化问题使用对应的函数（如 fminbnd, fmincon, linprog 等）即可，注意调整优化参数.

求解方程的根，最小二乘法作拟合也属于优化范畴，也可以使用 optimset 设置参数.

对于输出的结果，不仅需要看解是多少，而且需要看返回值 exitflag 与程序返回时的提示信息.

对于导数剧烈变化的函数，可以使用遗传算法求解. 此处由于代码量过大，且理解较为困难，不提供代码.MatLab 自带的 ga 需要精心设计参数才能得出较好结果.

优化问题可能单独出题，也可能穿插在任何应用中.

此外，二分法也是优化的方法之一. 对求根很有帮助.

6 迭代问题

迭代问题是比较困难的一类问题（特指函数迭代），这类问题没有固定的求解思路，但是仍然有一些加快编程速度的技巧.

- 使用匿名函数构造初始函数 $f_0(x)$. 这样操作的好处在于，可以方便的利用到之后的迭代中. 同时匿名函数表达分段函数非常方便，只需控制一下区间，写出对应的示性函数即可.
- 使用相同或相近的函数名表示迭代过程，如 $\phi = @(x)\phi(2x)+\phi(1-2x)$.
- 使用 fplot 绘图会优于 plot，因为 fplot 的绘制采用了自适应点密度的算法. 使用 plot 时，需要结合 arrayfun 计算函数值.
- 熟练使用符号变量与匿名函数的转化会提高速度.

以下是一个实例：

```
phi = @(x) ((-1 / 2 <= x) & (x < 1 / 2));
for i = 1: 10
    phi = @(x) (-(1 + sqrt(3)) * phi(2 * x) + (3 + sqrt(3)) *
    ↪  phi(2 * x - 1) + ...
          (3 - sqrt(3)) * phi(2 * x - 2) + (1 - sqrt(3)) *
          ↪  phi(2 * x - 3)) / 4;
end
fplot(phi, [0, 3], 'linewidth', 1);
```

迭代问题是数值分析的核心思想，对于函数的迭代可能是这一环节比较困难的部分，需要仔细理解.

对于点的迭代问题相对简单，但是使用匿名函数仍然能够提高效率.

7 杂项

编译出错，请先查询 MatLab 文档，再利用搜索引擎.

我们之前的数学实验文章都在这个网站上，可以看到大部分代码与报告：https://t0nyx1ang.github.io/Mathematical-Experiment

该网站将持续开启.

请尊重我们所有的文章版权，否则后果自负.