含参多项式系统的参数区间实根隔离

王兆吉

2018 年 3 月 20 日

1 问题描述

含参二元多项式系统 F 定义为：

$$\begin{cases} f_1(s,y,x,y)=0 \\ f_2(s,y,x,y)=0 \end{cases} \tag{1}$$

其中 f_1、f_2 是含参系数 (s,t) 的，关于变元 (x,y) 的多项式。

给定含参二元多项式系统 F 与变元区间 $[x',x'']\times[y',y'']$。若对任意的参数取值$(s_n,t_n)\in[s',s'']\times[t',t'']$，均存在$(x_n,y_n)\in[x',x'']\times[y',y'']$，使$(x_n,y_n)$构成系统的一组解，则称**($F$，$[x',x'']\times[y',y'']$) 在 $[s',s'']\times[t',t'']$ 上恒有解**。同理，若对任意的参数取值$(s_n,t_n)\in[s',s'']\times[t',t'']$，$F$ 在变元区间$[x',x'']\times[y',y'']$上均无解，则称**(F，$[x',x'']\times[y',y'']$) 在 $[s',s'']\times[t',t'']$ 上恒无解**。

对给定的含参系数多项式方程组 F、给定的参数区间 $[s',s'']\times[t',t'']$ 与变元区间 $[x',x'']\times[y',y'']$；**我们想要将参数区间 $[s',s'']\times[t',t'']$ 细分、判断、归类为"恒有解小区间"、"恒无解小区间"与"其他区间"。在输出参数区间不重叠的基础上，我们希望"其他区间"的测度小于给定值**。

基于上述讨论，我们需要设计两个函数：

1. *Test1*$(F,\square_{para},\square_{var})$

输入：含参二元多项式方程组 F、参数区间 $\square_{para}$、变元区间 $\square_{var}$

输出：若 (F，$\square_{var}$) 在 $\square_{para}$ 上恒有解，则返回 True

2. *Test2*$(\square_{para},\square_{var})$

输入：含参二元多项式方程组 F、参数区间 $\square_{para}$、变元区间 $\square_{var}$

输出：若 (F，$\square_{var}$) 在 $\square_{para}$ 上恒无解，则返回 True

对于给定的 F、$\square_{para}$ 与 $\square_{var}$，**我们希望根据输入找出一个阈值 ϵ 作终止条件：理想地，当 $\square'_{var}$ 的测度小于阈值 ϵ 时，如果 (F，$\square'_{var}$) 在 $\square_{para}$ 上恒有解，则必通过 *Test1*；如果 (F，$\square'_{var}$) 在 $\square_{para}$ 上恒无解，则必通过 *Test2***。换言之，如果细分 $\square_{var}$ 已经获得很小测度的 $\square'_{var}$ 却还未通过 *Test1* 与 *Test2*，那么我们可以认为：(F，$\square_{var}$) 在 $\square_{para}$ 上既不恒有解，也不恒无解，进而停止分割变元区间，转而对 $\square_{para}$ 进行一次细分，再依次判断。

Algorithm 1: 含参多项式系统的参数区间实根隔离算法

Input: 1.F 2.$\square_{para}$ 3.$\square_{var}$ 4.β

Output: $Q_{solvable}$、$Q_{unsolvable}$

Initialize queue $P_{para} \leftarrow \square_{para}$;
while $\beta < |\square_{para}| - \sum_{i=1}^{|Q_{solvable}|} |Q_{solvable_i}| - \sum_{i=1}^{|Q_{unsolvable}|} |Q_{unsolvable_i}|$ **do**
- $\tilde{\square}_{para} \leftarrow P_{para}.pop()$;
- Initialize stack $S_{var} \leftarrow \square_{var}$;
- flag=0;
- **while** $\min\limits_{\square'_{var} \in S} |\square'_{var}| \geq 0.5 \times \epsilon(\mathrm{F}, \square_{para}, \square_{var} : \mathrm{Test1}, \mathrm{Test2})$ **do**
  - $\tilde{\square}_{var} \leftarrow S_{var}.pop()$;
  - **if** $\mathrm{Test1}(\mathrm{F}, \tilde{\square}_{para}, \tilde{\square}_{var})$ *succeeds* **then**
    - $Q_{solvable}.push(\tilde{\square}_{para})$;
    - flag=1;
    - Break;
  - **if** $\mathrm{Test2}(\mathrm{F}, \tilde{\square}_{para}, \tilde{\square}_{var})$ *succeeds* **then**
    - **if** S_{var} *is empty* **then**
      - $Q_{unsolvable}.push(\tilde{\square}_{para})$;
      - flag=1;
      - Break;
  - **else**
    - Split $\tilde{\square}_{var}$ into 2^n congruent subboxes, and add them to S_{var};
- **if** *flag==0* **then**
  - Split $\tilde{\square}_{para}$ into 2^n congruent subboxes, and add them to P_{para};

2 需要解决的问题

当前能够解决对二元多项式系统的实根隔离。即在系数不含参数区间时，我们能通过 $MK\ Test$ 与 $Exclusion\ test$ 对多项式系统实根解个数判断。
当处理含参系统1时，由于系数所在位置不固定，需要解决的第一个问题就是：

如何将 $MK\ Test$ 推广为参数是区间的 $Test1$？
如何将 $Exclusion\ test$ 推广为参数是区间的 $Test2$?

直觉上，针对不同的多项式系统 F、*Test1*、*Test2*、$\square_{para}$ 与 $\square_{var}$，细分后变元盒子应该会有不同的阀值 ϵ，待解决的第二个问题就是是：

如何计算算法 1 中提及的 $\epsilon(\boldsymbol{F}, \square_{para}, \square_{var} : \textit{\textbf{Test1}}, \textit{\textbf{Test2}})$

3 一个例子

含参二元多项式系统不易想象，我们可以从含参一元多项式等式入手。
考虑含参 s 的关于 w 一元多项式等式 F:

$$s \times w = 0$$

若考虑以 $\square_{var} = [-2, 10]$、$\beta = 0.01$、$\square_{para} = [-3, 3]$ 作为参数输入算法1后计算，不难发现在迭代第一步 $\square_{para}$ 时即可通过 $Test1$ <u>（此处仅是假设$Test1$是（F，$\square_{var}$）在$\square_{para}$恒有解的充要条件。即便不是充要条件，$\square_{para}$ 分割后的每一个区间均为（F，$\square_{var}$）的恒有解区间，这是因为$0 \in \square_{var}$。所以在保证$Test1$的有效性与正确性前提下，皆能通过$Test1$）</u>，一步即终止（或多步终止，但输出区间构成 $[-3, 3]$ 的闭覆盖），输出 $[-3, 3]$。

但是当输入 $\square_{var} = [2, 10]$、$\beta = 0.01$、$\square_{para} = [-3, 3]$ 时，在迭代第一步 $\square_{para}$ 时算法无法终止。这是因为 $[0, 3]$ 混有恒有解点 $[0, 0]$。若不设置终止条件 $\epsilon(F, \square_{para}, \square_{var} : \textit{Test1}, \textit{Test2})$，算法将构成死循环。当选取适当的 ϵ 时，内层的栈结构会进行深度优先遍历，也就是说，程序至多进行 $\mathrm{ceil}(log_{1/2}(epsilon))$ 次迭代就会终止，从这里可以看出这是很有效率的算法。

事实上，我们并不知道 $Test1$、$Test2$ 需要的条件究竟有多强，或是否能够弱到成为充要条件；但从某种角度来讲，条件越宽泛的 $Test1$、$Test2$ 越易满足，也更难设计，对应的 ϵ 也就相对地越大。