Firstly, thanks very much for reviewers' commonts and suggestions.

1) We have corrected some of english problems by reviewers commonts.

2) We given an example to demonstrate the theorems and algorithms.

3) We add some word description parts to expression the theorems and algorithms.

4) We separate the theorems and algorithms into two sections.

5) We simplifies the descriptions of algorithms.

6) We add some noations in proof of theorem 5, make it more clear.

We list the response for reviewers' questions in the following:

1) The paper says that logcf is much more efficient than other tools. But it is unclear why it is more efficient.

Is it mainly because of the improved root bound or because you have better implementation techniques?

We think the main reason is that improved root bound and

we have prove that our bound is more sharp than other bounds

and its computational complexity is also acceptable.

Is the improvement related to particular types of polynomials?

In experiment section, we test logcf in different type of polynomials, from the test results there the improvement does not relate to particular types of polynomials.

    Have you found the cases when your method is less efficient than state of art implementations and why?

We find that logcf is slower than RootIntervals and realroot in WPn（ ） polynomials. We plan to analysis in next work.I

It will be useful to demonstrate the bounds computed by your methods and other methods by benchmark examples.

We add an example to demonstrate it.

If in your implementation, you use the same bound by other tools,  how efficient will it be w.r.t. other tools?

In next work, we will give more experiments about real root isolation.

2) for random generated examples, you did not mention the size of the polynomials.

We add the generate method about random cases.

Algorithm 6, step 2, is lessOne(p) is false, why the root bound is 2?

“2” is just a label to indicate that the upper bound is greater than “1” and we do not use the concreate value of it.

Algorithm 6, line 9, the use of loop condition i==n is weird, you mean while true, since at the beginning of

      each iteration, the value of i is for use n. Line 25, should you return 2^(base-1)?

Loop condition i==n is not weird, as “i” is reseted to “n” under condition j==lastNeg-2.

The return is also right since we convert division operator by multiplication operation in this algorithm for improve efficiency.

In this case, p(x)🡪 2^np(x).

Corollary, you can not say "it cost at most ...." if you did not specify how you compute 2^({n-j}\*base).

For more correction, we change the words to “O(n\log(u+1)) addition and multiplication symbolic operators”.