Project report (MAN-300) on

Factorization of polynomials using Rational Root Test

Devendra Kumar (13312005) MSM-III

Department of Mathematics, IIT Roorkee

Under supervision of

Dr. Maheshanand

Department of Mathematics, IIT Roorkee

Submitted to:

Department of Mathematics, IIT Roorkee

Roorkee, Haridwar (Uttarakhand)-247667

Academic year: Spring Semester 2015-16

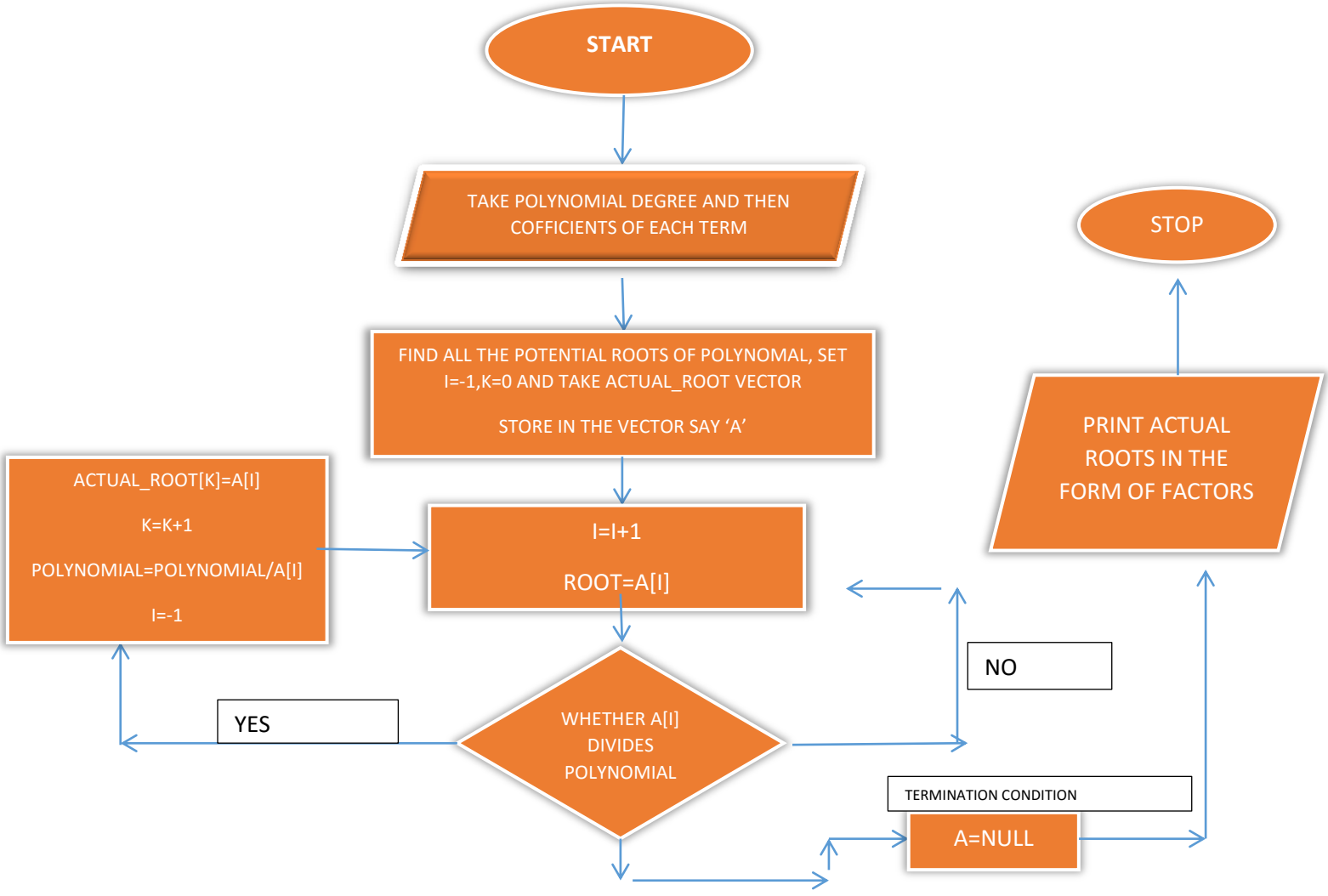
Acknowledgement:

I thank to my guide Dr. Maheshanand to encourage me to opt for this project. I thank him for the nascent intuition he placed inside me to work on such a great problem. I also thank purplepage.com and Wikipedia for the background content about various theorems.

Abstract:

There are various applications like MATLAB, Maple etc. These applications are capable of factoring polynomial in real time.
The problem is that they are not open source; they are not available for free use.
I, under the proper support and guidance of Dr. Maheshanand, am trying to create an open source algorithm which can factorize a polynomial with only real rational roots with any multiplicity.
Algorithm test the potential roots to satisfy the given polynomial. If it is satisfied then the test potential root is actual root of that polynomial.

Algorithm flowchart:



Time complexity:

Synthetic division for division of polynomial by rational root costs: O(n); n→ no. of terms in polynomial.

Factor function have complexity in worst case as O(m); m→ no. m.

Rational_root function has complexity of O(xy); x→no. of factors in constant term, y→no. of factors in leading coefficient.

Algorithm complexity→O(max{m,n,xyk}); k→ sum of all multiplicities

Future Expectations:

Since the algorithm is open and free to use, modify, copy and reuse. It will help many students and researchers to develop this algorithm as per their needs and use to solve other polynomial problems.

Codes:

```
#include<iostream>

#include<vector>  // used for vector<>

#include<math.h> // used for abs()


using namespace std;


//structure to hold the division result

struct division_result{

        vector<float> quotient;

        float remainder;

};
```

```
//synthetic division function  complexity O(n=degree of poly)
division_result synthetic_division(vector<float>cofficient, float root){
		float rem=0;
		vector<float>poly_coff;

		for(std::vector<float>::iterator i=cofficient.begin();i!=cofficient.end();i++){
			rem=rem*root+(*i);
			poly_coff.push_back(rem);
			//poly_coff.pop_back();
			cout<<rem<<"  ";
		}
		division_result result;
		result.remainder=rem;
		poly_coff.pop_back();
		result.quotient=poly_coff;

		return result;
}
//end of synthetic division function

//calculation of factors
vector<float> factors(int f){
  vector<float>fact;
  for(int i=1;i<=abs(f);++i)
   {
      if(f%i==0)
        fact.push_back(i);
```

```
        }
        return fact;
}
//end of factors

//calculation of all rational roots start
vector<float> rational_roots(vector<float> poly){

        int leading_coff = poly.front();
        int constant_term = poly.back();

        cout<<"Factors of leading coff: ";
        vector<float> leading_coff_factors= factors(leading_coff);
        for(std::vector<float>::iterator i=leading_coff_factors.begin();i!=leading_coff_factors.end();i++)
        cout<<*i<<"   ";

        cout<<"\nFactors of constant term: ";
        vector<float> constant_term_factors= factors(constant_term);
        for(std::vector<float>::iterator j=constant_term_factors.begin();j!=constant_term_factors.end();j++)
        cout<<*j<<"   ";

        vector<float> rational_roots;
        for(std::vector<float>::iterator i=leading_coff_factors.begin();i!=leading_coff_factors.end();i++){
                for(std::vector<float>::iterator j=constant_term_factors.begin();j!=constant_term_factors.end();j++){
                        rational_roots.push_back((*j)/(*i));
                        rational_roots.push_back(-(*j)/(*i));
                }
```

```
        }
        return rational_roots;
}
//end of rational_roots


//testing
int main(){
        vector<float>factor_of_polynomial;
        cout<<"Testing Synthetic division:\n";
        vector<float> fop;
        division_result res;
        cout<<"Degree of Polynomial:";
        int deg;
        cin>>deg;
        vector<float>poly;
        float temp;
        cout<<"cofficient of polynomial (n+1) terms,leading cofficient term first:";
        while(deg+1){
                cin>>temp;
                poly.push_back(temp);
                deg--;
        }



        fop= rational_roots(poly);
        cout<<"\nRational Roots:";
        for(std::vector<float>::iterator i=fop.begin();i!=fop.end();i++)
```

```
                cout<<*i<<"  ";
        for(std::vector<float>::iterator k=fop.begin();k!=fop.end();k++){
                cout<<"\ngoing with "<<*k<<"\n.....\n";
        res= synthetic_division(poly,*k);
        cout<<"\nremainder:  "<<res.remainder<<"\nQuotient: ";

        if(res.remainder==0){
                factor_of_polynomial.push_back(*k);
                poly=res.quotient; //// solved:: multiplicity condition is checked
                k=fop.begin();
        }
        for(std::vector<float>::iterator i=res.quotient.begin();i!=res.quotient.end();i++){
                cout<<*i<<"  ";
        }
        }
        cout<<"\n\n**********\nfactorization: ";
        for(std::vector<float>::iterator g=factor_of_polynomial.begin();g!=factor_of_polynomial.end();g++)
        cout<<"(x"<<showpos<<-*g<<")";
}
```

References:

1. Wikipedia: https://en.wikipedia.org/wiki/Factor_theorem (24th feb 2016)
2. Wikipedia: https://en.wikipedia.org/wiki/Synthetic_division (24th feb 2016)
3. Wikipedia: https://en.wikipedia.org/wiki/Talk%3AFactorization_of_polynomials#Kronecker.27s_method (27th feb 2016)
4. Wikipedia: https://en.wikipedia.org/wiki/Factorization_of_polynomials (27th feb 2016)
5. Gutenberg.us: http://www.gutenberg.us/articles/kronecker's_method (28th feb 2016)
6. Purplemath.com: http://www.purplemath.com/modules/rtnlroot.html (29th feb 2016)
7. Computation in Mathematics by Carl and vhensile

NO