Demonstration Examples

Jonathan D. Hauenstein

1. Illustrative example

For an illustrative first example of using `Bertini` [1], we solve the following quintic polynomial equation which can not be solved in terms of radicals:

$$f(x) = x^5 - x + 1 = 0. \quad (1)$$

The following `input` file can be used to solve (1):

```
CONFIG
END;
INPUT
 variable_group x;
 function f;

 f = x^5 - x + 1;
END;
```

Executing `Bertini` from the command line:

```
 >> bertini input
```

tracks $\deg f = 5$ paths to compute the 5 solutions to (1). A portion of the screen output is:

```
  Multiplicity  |  Number of real solns |  Number of non-real solns
------------------------------------------------------------------------
       1        |       1               |       4
```

showing that (1) has 1 real and 4 non-real solutions. In our test, the following is the output file `finite_solutions`:

```
5

7.648844336005846e-01 3.524715460317263e-01

-1.812324444698754e-01 1.083954101317711e+00

-1.167303978261419e+00 -1.665334536937735e-16

7.648844336005847e-01 -3.524715460317264e-01

-1.812324444698754e-01 -1.083954101317711e+00
```

which first lists the number of solutions (5) and then numerical approximations of the real and imaginary coordinates of the solutions, i.e., the line

```
7.648844336005846e-01 3.524715460317263e-01
```

corresponds with

$$0.7648844336005846 + 0.3524715460317263 \cdot \sqrt{-1}.$$

Hence, the third point listed above in `finite_solutions` is a numerical approximation of the real solution of (1). The file `main_data` provides information about the quality of the numerical approximations of each solution. For example, here is a portion of `main_data` from our test:

```
Solution 3 (path number 0)
Estimated condition number: 5.940070457750704e+01
Function residual: 3.972054645195637e-15
Latest Newton residual: 4.405327952961146e-16
T value at final sample point: 3.906250000000000e-04
Maximum precision utilized: 52
T value of first precision increase: 0.000000000000000e+00
Accuracy estimate, internal coordinates (difference of last two endpoint estimates):  5.083739718695013e-13
Accuracy estimate, user's coordinates (after dehomogenization, if applicable):  5.679652284963224e-13
Cycle number: 1
7.648844336005846e-01 3.524715460317263e-01
Paths with the same endpoint, to the prescribed tolerance:
Multiplicity: 1
```

2. SHARPEN ILLUSTRATIVE EXAMPLE

To demonstrate the ability of `Bertini` to compute solutions to arbitrary accuracy, we utilize the sharpening module on the illustrative quintic polynomial equation (1) to refine the solutions to 30 digits using Newton's method:

```
CONFIG
 SharpenDigits: 30;
END;
INPUT
 variable_group x;
 function f;

 f = x^5 - x + 1;
END;
```

In our test, the unique real solution is listed in the the output file `real_finite_solutions` is:

```
-0.116730397826141868425604589985484218O724e1 0.367341984631964846240230167881951774318 3e-39
```

which shows that the imaginary part of the numerical approximation is on the order of 10^{-39}.

3. CERTIFY ILLUSTRATIVE EXAMPLE

We now demonstrate using `alphaCertified` [3] to prove the output of `Bertini` from above for solving (1). The polynomial is described based on monomials in the following `polySys` file:

```
1 1

3
5 1 0
1 -1 0
0 1 0
```

The first line states that there is one variable and one polynomial. The next line states that the first polynomial has 3 monomials. Each monomial is written by listing the degrees of the variables and then real and imaginary parts of the coefficient. That is, the last three lines correspond to x^5, $-x$, and 1, respectively. This file can be created using the `alphaCertifiedMaple` library.

If we want to use exact rational certification in `alphaCertified`, we need to write the real and imaginary coordinates of the points using rational numbers. For example, we use the following `points` file (which can also be created using the `alphaCertifiedMaple` library):

```
5

-116/100 0

764/1000 -352/1000

764/1000 352/1000

-181/1000 1083/1000

-181/1000 -1083/1000
```

The execution `alphaCertified` from the command line:

```
>> alphaCertified polySys points
```

produces the following screen output:

```
Analyzing 5 points using exact arithmetic.

Isolating 5 approximate solutions.

Classifying 5 distinct approximate solutions.

Rational certification results:

Number of points tested:              5
Certified approximate solutions:      5
Certified distinct solutions:         5
Certified real distinct solutions:    1
```

This shows that all 5 points tested correspond with distinct solutions to (1) and exactly one of the solutions is real.

4. Multihomogeneous example

For a matrix $A \in \mathbb{C}^{N\times N}$, $\lambda \in \mathbb{C}$ is an *eigenvalue* of A with corresponding *eigenvector* $v \in \mathbb{C}^N$ if

$$Av - \lambda v = 0 \tag{2}$$

and $v \neq 0$. We utilize this example to demonstrate using `Bertini` to solve systems defined on a product of affine and projective space. Since (2) is homogeneous with respect to v, i.e., if v is an eigenvector of A corresponding to λ, then αv is also an eigenvector of A corresponding to λ for any $\alpha \neq 0$, we should naturally treat (2) as a system of N equations defined on the product space $\mathbb{C} \times \mathbb{P}^{N-1}$. In particular, since each equation in (2) is linear in both λ and v, a homotopy on $\mathbb{C} \times \mathbb{P}^{N-1}$ to solve (2) requires tracking $\binom{N}{1} = N$ solution paths (see [2, § 5.1]), which is equal to the generic number of distinct eigenvalue and eigenvector pairs.

For example, for the matrix

$$A = \begin{bmatrix} 1 & 2 \\ 3 & 4 \end{bmatrix}$$

the following `input` file computes eigenvalue and eigenvector pairs:

```
CONFIG
END;
INPUT
 variable_group l; % eigenvalue
 hom_variable_group v1,v2; % eigenvector
 function f1,f2;
 constant a11,a12,a21,a22; % matrix
 a11 = 1; a12 = 2; a21 = 3; a22 = 4;

 f1 = a11*v1 + a12*v2 - l*v1;
 f2 = a21*v1 + a22*v2 - l*v2;
END;
```

Executing `Bertini` tracks two paths. The following is `finite_solutions` from our test:

```
2

5.372281323269013e+00 0.000000000000000e+00
4.574271077563382e-01 -2.775557561562891e-17
1.000000000000000e+00 0.000000000000000e+00

-3.722813232690144e-01 0.000000000000000e+00
1.000000000000000e+00 0.000000000000000e+00
-6.861406616345070e-01 5.551115123125783e-17
```

This shows the eigenvalues of A are approximately 5.372 and -0.372. Notice that the eigenvectors are scaled so that largest coordinate (in absolute value) is 1.

5. Numerical irreducible decomposition

To illustrate computing a numerical irreducible decomposition using `Bertini`, we consider the following polynomial system defined on $\mathbb{C}^6$:

$$f(x) = \begin{bmatrix} x_1x_5 - x_2x_4 \\ x_2x_6 - x_3x_5 \end{bmatrix}. \tag{3}$$

This is accomplished using the following `input`:

```
CONFIG
 TrackType: 1; % compute numerical irreducible decomposition
END;
INPUT
 variable_group x1,x2,x3,x4,x5,x6;
 function f1,f2;
```

```
 f1 = x1*x5 - x2*x4;
 f2 = x2*x6 - x3*x5;
END;
```

A portion of the screen output from executing `Bertini` is:

```
************** Decomposition by Degree **************

Dimension 4: 2 classified components
------------------------------------------------------
   degree 1: 1 component
   degree 3: 1 component

******************************************************
```

showing that (3) defines two irreducible components of dimension 4, one has degree 1 and the other has degree 3.

In our test, a portion of `main_data` is:

```
Number of variables: 6
Variables:  x1 x2 x3 x4 x5 x6
Rank: 2

----------DIMENSION 4----------

NONSINGULAR SOLUTIONS
---------------
Path number: 3
Component number: 0
Estimated condition number: 1.000572e+01
7.950394424236318e-01 1.008831895707915e+00
0.000000000000000e+00 0.000000000000000e+00
-4.302450583776669e-01 9.066401487298040e-01
1.810287256420423e-01 -1.095072346789127e+00
2.759195389687779e-17 3.102179290555893e-18
-2.160905745563312e-01 6.033840888172206e-01
Multiplicity: 1
Deflations needed: 0
```

which shows, for example, that the x_2 and x_5 coordinates are zero on this component.

The file `witness_data` contains the data required to use the witness set for further computations.

6. Sampling

Using the witness set computed when solving (3) stored in `witness_data`, we next use `Bertini` to sample an irreducible component. The modified `input` file is:

```
CONFIG
 TrackType: 2; % sample
END;
```

```
INPUT
 variable_group x1,x2,x3,x4,x5,x6;
 function f1,f2;

 f1 = x1*x5 - x2*x4;
 f2 = x2*x6 - x3*x5;
END;
```

Executing `Bertini` produces a sequence of menus for selecting the dimension, component, number of points, output type (screen or file), and file name [if necessary]. For example, in our test, sampling 3 points on linear component yields the following three points:

```
3

5.920239720895480e-01 -4.418341177482604e-01
-3.658401632350801e-22 3.530846845538152e-19
3.134261347213816e+00 1.187159473313852e+00
-5.800140329537541e-01 -2.618666107204265e+00
1.762472291907067e-20 -5.909247530626338e-20
-4.990593900777348e-01 -1.154504588134285e+00

9.465018523915929e-01 -2.510458828984211e+00
-1.304063261824207e-18 -1.447731669164072e-18
2.583294298316470e+00 1.808135524193481e+00
1.548253600150566e+00 2.248756149691759e+00
-6.960997818938286e-19 1.521255718232070e-18
-2.624002809603148e+00 -4.019203985948467e+00

-4.603118770899699e-02 4.026715942103231e-02
-7.430492710325136e-21 -4.579109411165913e-22
1.617409479899316e+00 4.897963058450606e-01
-3.155419868887653e-01 -1.398760129469327e+00
-7.497965781365959e-21 -2.124147634696427e-20
-1.512720350616142e-01 -8.294039396795403e-01
```

which again shows that the x_2 and x_5 coordinates are zero on this component.

7. PROJECTION

Due to time constraints, our final computation is to project the degree 3 component of (3) onto the x_1, x_3, x_4, x_6 variables. We utilize `witness_data` together with the following `input` file:

```
CONFIG
 TrackType: 5; % projection
END;
INPUT
 variable_group x1,x2,x3,x4,x5,x6;
 function f1,f2;
```

```
 f1 = x1*x5 - x2*x4;
 f2 = x2*x6 - x3*x5;
END;
```

To define the projection, we create the file `projection` which provides a boolean for each variable depending on if that coordinate is in the image. Projecting onto the x_1, x_3, x_4, x_6 variables is described by

```
1 0 1 1 0 1
```

Executing `Bertini` produces a sequence of menus for selecting the dimension and component. Projecting the degree 3 component produces the following screen output:

```
Dimensions
  Projection: 3
       Fiber: 1

Degrees
  Projection: 2
       Fiber: 1
```

showing that the image has dimension 3 and degree 2 while the fiber has dimension 1 and degree 1. In fact, the image is a hypersurface defined by $x_1x_6 - x_3x_4 = 0$.

References

[1] D.J. Bates, J.D. Hauenstein, A.J. Sommese, and C.W. Wampler. Bertini: Software for Numerical Algebraic Geometry. Available at `bertini.nd.edu`.

[2] D.J. Bates, J.D. Hauenstein, A.J. Sommese, and C.W. Wampler. *Numerically Solving Polynomial Systems with Bertini*, Volume 25 of Software, Environments, and Tools, SIAM, Philadelphia, 2013.

[3] J.D. Hauenstein and F. Sottile. Algorithm 921: alphaCertified: certifying solutions to polynomial systems. *ACM Trans. Math. Software*, 38(4), 28, 2012.