alphaCertified

Jonathan D. Hauenstein

September 16, 2011

alphaCertified

1 Introduction to alphaCertified

The program alphaCertified by Jonathan D. Hauenstein and Frank Sottile implements algorithms based on Smale's α-theory to certify solutions to polynomial and polynomial-exponential systems. This manual provides detailed instructions on how to use alphaCertified while [4, 5] provides more information regarding the mathematical theory underlying alphaCertified.

2 Compiling alphaCertified

The program alphaCertified is written in C and uses the GMP[3] and MPFR[2] libraries to perform rational and arbitrary floating point arithmetic. To compile alphaCertified, the user needs to verify the settings in `Makefile` associated with the C compiler and the location of these libraries. The following is an example of the first three lines in `Makefile` which specifies using `gcc` along with the locations of the GMP and MPFR installation directories.

```
COMP=gcc
GMP=/home/GMP_4_3_2/
MPFR=/home/MPFR_2_4_2/
```

The following is the next two lines in `Makefile` which specifies to the C compiler which libraries to link to and location of the header files.

```
LIB=-lm -L$(MPFR)/lib/ -lmpfr -L$(GMP)/lib/ -lgmp
INC=-I$(MPFR)/include -I$(GMP)/include
```

When using `gcc`, for example, if the proper static libraries were created, adding the "-static" option on the LIB line, displayed below, will create a statically linked executable.

```
LIB=-static -lm -L$(MPFR)/lib/ -lmpfr -L$(GMP)/lib/ -lgmp
```

Once `Makefile` is setup, simply run 'make' to compile alphaCertified.

3 Using alphaCertified

To use alphaCertified, the user needs to create at least two files which specify the polynomial or polynomial-exponential system and the points to test. An optional third file can be used to adjust the configuration settings. See Appendix A for a detailed description of each configuration setting.

3.1 Polynomial systems

A polynomial system is entered into a file by listing the monomials and the coefficients appearing in each polynomial. The system must be square or overdetermined and each coefficient must be a complex rational number.

The first line of the file lists both the number of variables and the number of polynomials for the polynomial system. Then, the file contains a block for each polynomial which contains the number of terms followed by the degrees of each variable in the monomial and the real and imaginary parts of its coefficient. For example, if the polynomial system depends upon three variables, say x, y, and z, the term

$$\left(\frac{1}{2}+3i\right)xy^2z$$

would be written as

$$1 \quad 2 \quad 1 \quad 1/2 \quad 3$$

since the degrees of x, y, and z in xy^2z are 1, 2, and 1, respectively, and the coefficient has real part $\frac{1}{2}$ and imaginary part 3.

For a complete example, consider $f(x,y,z) = \begin{bmatrix} x^2-3xy+z^4 \\ \left(\frac{1}{2}+3i\right)xy^2z-9z+2x+7 \\ z^{10}-i \end{bmatrix}$. The polynomial system file for f is as follows.

3	3			
3				
2	0	0	1	0
1	1	0	−3	0
0	0	4	1	0
4				
1	2	1	1/2	3
0	0	1	−9	0
1	0	0	2	0
0	0	0	7	0
2				
0	0	10	1	0
0	0	0	0	−1

Table 1: Example of a polynomial system file for f

3.2 Polynomial-exponential systems

Starting with version 1.2, alphaCertified implements the algorithms of [5] for certifying solutions to square systems of polynomial-exponential functions. The polynomial-exponential system must be of the form

$$\begin{bmatrix} P_i(x_1,\ldots,x_n,y_1,\ldots,z_m), & i=1,\ldots,n \\ y_i - g_i(\beta_i x_{\sigma_i}), & i=1,\ldots,m \end{bmatrix}$$

where each P_i is a polynomial with complex rational coefficients, β_i is a complex rational number, and $g_i(z)$ is either $\exp(z)$, $\sin(z)$, $\cos(z)$, $\sinh(z)$, or $\cosh(z)$. Moreover, polynomial-exponential certification must use floating point arithmetic (see Appendix A for more details).

The first line of the file lists both the number of variables, e.g., $n+m$, and the number of polynomials in the polynomial-exponential system, e.g., n. Then, the file contains a block for each polynomial following the structure described in Section 3.1. The final block contains a line for each additional function which lists the integer σ_i, a string describing the function g_i, and the real and imaginary parts of β_i. The following table lists the strings for the possible functions.

function	string
$\exp(z)$	X
$\sin(z)$	S
$\cos(z)$	C
$\sinh(z)$	SH
$\cosh(z)$	CH

Table 2: Functions and strings

For a complete example, consider $f(x,y,z) = \begin{bmatrix} x^2 - 3xy + z^4 \\ \left(\frac{1}{2}+3i\right)xy^2z - 9z + 2x + 7 \\ z - \cos((3-4i)y) \end{bmatrix}$. The polynomial-exponential system file for f is as follows.

```
3 2

3
2 0 0 1    0
1 1 0 −3   0
0 0 4 1    0
4
1 2 1 1/2  3
0 0 1 −9   0
1 0 0 2    0
0 0 0 7    0

2 C 3 −4
```

Table 3: Example of a polynomial-exponential system file for f

3.3 Input points

The points are read into alphaCertified using the requested arithmetic type (see Appendix A for more details). In particular, if alphaCertified is configured to use rational arithmetic, the real and imaginary parts of the coordinates for each point must be a rational number. Likewise, if alphaCertified is configured to use P-bit floating point arithmetic, the the real and imaginary parts of the coordinates for each point must be a floating point number and are read into alphaCertified using P-bit precision. When using floating point arithmetic, many of the output files generated by Bertini [1] can be used for input into alphaCertified.

The first line of the file lists the number of points in the file. This is followed by a block for each point which contains the real and imaginary parts of each coordinate. For example, the following presents

$$S = \left\{ \begin{bmatrix} \frac{2}{3} + 2i \\ -7 \\ 3i \end{bmatrix}, \begin{bmatrix} \frac{1}{8} \\ -7 + 2i \\ -1 - \frac{2}{5}i \end{bmatrix} \right\}$$

using rational and 10-digit floating point representation.

```
2

2/3   2
−7    0
0     3

1/8   0
−7    2
−1    −2/5
```

Table 4: Example using rational numbers for S

```
2

0.6666666667   2
−7             0
0              3

0.125          0
−7             2
−1             −0.4
```

Table 5: Example using floating point numbers for S

3.4 Configuration settings

To adjust the configuration settings, the user needs to create a file listing the configurations along with the requested values. See Appendix A for each configurable setting and its acceptable values.

For example, the following presents settings that would only certify approximate solutions using 192-bit floating point precision.

ALGORITHM: 0; ARITHMETICTYPE: 1; PRECISION: 192;

Table 6: Example of configuration settings

3.5 Running alphaCertified

The command line arguments for alphaCertified specify the names of the files to use. The default names for the corresponding files are `polynomialSystem`, `testPoints`, and `settings`.

The command line arguments correspond to the names of these files in order. For example, if there are two arguments, the first is the name of the system file and the second is the name of the file containing the points to test. Default names will be used when less than three command line arguments are used. Errors will be returned if either the system or the test points file do not exist. If the configuration settings file does not exist, the default settings will be used.

For example, if there is no file named `settings` in the current folder, the following Linux command runs alphaCertified using the default settings on the system file named `polySys` and the test points file named `points`.

$$\gg \text{./alphaCertified polySys points}$$

4 Output of alphaCertified

The output of alphaCertified is a collection of files that are dependent upon the configuration settings and the polynomial system as well as an onscreen summary. The following lists the possible files created by alphaCertified, their format, and the settings needed to create them. See Appendix A for more details regarding the configurations settings.

- `approxSolns`

  This file, in the format of the input points files described in Section 3.3, lists the points which are certifiably approximate solutions. This file is always created.

- `constantValues`

  This file lists an upper bound of α, an approximation of β, and an upper bound of γ for each point (see [4] for details on how these values are computed). These values are always printed using a 16-digit floating point representation. The format is similar to that presented in Table 5. In particular, the first line lists the number of points and then a block for each point which lists the computed values for α, β, and γ on separate lines. This file is always created.

- `distinctSolns`

  This file, in the format of the input points files described in Section 3.3, lists the points which are certifiably approximate solutions that correspond to distinct solutions. This file is created when ALGORITHM is at least 1.

- `isApproxSoln`

  This file lists boolean values (0 or 1) describing whether each point has been certified to be an approximate solution. The first line lists the number of points and then the boolean value is listed for each point. Note that a value of 0 means that alphaCertified was not able to certify that it was an approximate solution. This file is always created.

- `isDistinctSoln`

  This file, in the same format as `isApproxSoln`, lists values describing whether each point corresponds to a distinct solution. A value of -2 means that the point was not able to be certified as an approximate solution. A value of -1 means that the point corresponds to a solution that is distinct from the certifiably approximate solutions that precede it in the list of points. A nonnegative value, say j, means that this point and the j^{th} point correspond to the same solution, where the points are numbered staring with 0. This file is created when ALGORITHM is at least 1.

- `isRealSoln`

  This file, in the same format as `isApproxSoln`, lists values describing whether each point corresponds to a real solution. A value of -2 means that the point was not able to be certified as an approximate solution. Otherwise, the value is a boolean value (0 or 1) describing if the point corresponds to a real solution. This file is created when ALGORITHM is 2 and the polynomial system is real.

- `nonrealDistinctSolns`

  This file, in the format of the input points files described in Section 3.3, lists the points which are certifiably approximate solutions that correspond to distinct nonreal solutions. This file is created when ALGORITHM is 2 and the polynomial system is real.

- `realDistinctSolns`

  This file, in the format of the input points files described in Section 3.3, lists the points which are certifiably approximate solutions that correspond to distinct real solutions. This file is created when ALGORITHM is 2 and the polynomial system is real.

- `redundantSolns`

  This file, in the format of the input points files described in Section 3.3, lists the points which are certifiably approximate solutions and correspond to the same solution as another certifiably approximate solution that precedes it in the list of points. This file is created when ALGORITHM is at least 1.

- `refinedPoints`

This file, in the format of the input points files described in Section 3.3, lists the most accurate internally computed approximation of the corresponding solution. If REFINEDIGITS is positive, say τ, then, for each certifiable approximate solution, the point listed in this file is within $10^{-\tau}$ of the corresponding solution. This file is always created.

- `summary`

  This file is a human-readable summary for each point. The first part of this file reprints the onscreen summary of the results. This is followed by a block for each point which lists the point, the results for that point, and the computed values of α, β, and γ for both the original point and its corresponding point printed in `refinedSolns` (see [4] for details on how these values are computed). The last part of this file contains configuration settings and version information for alphaCertified. This file is always created.

- `unknownPoints`

  This file, in the format of the input points files described in Section 3.3, lists the points which can not be certified as approximate solutions. This file is always created.

5 Performing Newton iterations using alphaCertified

When using alphaCertified for certifying solutions, the certified approximate solutions can be refined using Newton's method to any given accuracy using the REFINEDIGITS configuration setting. Instead of only refining the certified approximate solutions, alphaCertified can also be used to perform Newton iterations on all of the input points using the NEWTONONLY and NUMITERATIONS configuration settings. See Appendix A for more details regarding these configuration settings.

Note that if floating point arithmetic is begin used, i.e., ARITHMETICTYPE is 1, the internal working precision is automatically increased during each iteration.

6 Maple interface for alphaCertified

The Maple interface for alphaCertified can be used to construct the input files, run alphaCertified, and read in output files. After updating `libname` to include the folder where the alphaCertified Maple interface is located, it can be loaded with the following command:

$$> \ with(alphaCertifiedMaple);$$

yielding the following output:

$$[alphaCertified, alphaCertifiedExp, defaultExpSettings, defaultSettings, loadOutput, printPoints, printPolyExpSystem, printPolynomialSystem, printSettings]$$

The following describes how to use these nine procedures.

6.1 alphaCertified

The Maple procedure alphaCertified constructs the input files for a polynomial system, runs alphaCertified, and loads output data.

PointsData := alphaCertified(alphaPath, Polys, Vars, Points, Settings);

- `alphaPath`

  A string which is the path to an alphaCertified executable file.

- `Polys`

  A list containing the polynomial system.

- `Vars`

  A list containing the variables of the polynomial system.

- `Points`

  A two-dimensional list containing the points to test.

- `Settings (optional)`

  A Maple record containing the configuration settings. See Section 6.4 for more details constructing this record.

- `PointsData`

  A Maple record containing output data from alphaCertified. The fields in this record are:

  - `alpha`
    A list containing an upper bound of α at each point.
  - `beta`
    A list containing an approximation of β at each point.
  - `gamma`
    A list containing an upper bound of γ at each point.
  - `refinedPts`
    A two-dimensional list containing the refined points.
  - `isApproxSoln`
    A list that describes if each point is a certifiable approximate solution. Each value is either "Unknown", "Yes", or "No".
  - `isDistinctSoln`
    A list that describes if each point corresponds to a distinct solution. Each value is either "Unknown", "Yes", or "No".
  - `isRealSoln`
    A list that describes if each point corresponds to a real solution. Each value is either "Unknown", "Yes", or "No".

Example

```
> alphaPath := "./alphaCertified";
> Polys := [x * y^2 + x, x * y - y];
> Vars := [x, y];
> Points := [[0,0],[1,1+I],[1,-I]];
> PointsData := alphaCertified(alphaPath, Polys, Vars, Points);
```

6.2 alphaCertifiedExp

The Maple procedure alphaCertified constructs the input files for a polynomial-exponential system, runs alphaCertified, and loads output data.

PointsData := alphaCertifiedExp(alphaPath, Funcs, Vars, Points, Settings);

- `alphaPath`

  A string which is the path to an alphaCertified executable file.

- `Funcs`

  A list containing the polynomial-exponential system.

- `Vars`

  A list containing the variables of the polynomial-exponential system.

- `Points`

  A two-dimensional list containing the points to test.

- `Settings (optional)`

  A Maple record containing the configuration settings. See Section 6.4 for more details constructing this record.

- `PointsData`

  A Maple record containing output data from alphaCertified. The fields in this record are:

  - `alpha`
    A list containing an upper bound of α at each point.
  - `beta`
    A list containing an approximation of β at each point.
  - `gamma`
    A list containing an upper bound of γ at each point.
  - `refinedPts`
    A two-dimensional list containing the refined points.

- `isApproxSoln`

  A list that describes if each point is a certifiable approximate solution. Each value is either "Unknown", "Yes", or "No".

- `isDistinctSoln`

  A list that describes if each point corresponds to a distinct solution. Each value is either "Unknown", "Yes", or "No".

- `isRealSoln`

  A list that describes if each point corresponds to a real solution. Each value is either "Unknown", "Yes", or "No".

Example

```
> alphaPath := "./alphaCertified";
> Funcs := [x * y^2 + x, y - exp(3 * x)];
> Vars := [x, y];
> Points := [[0,1],[2,1+I]];
> PointsData := alphaCertifiedExp(alphaPath, Funcs, Vars, Points);
```

6.3 defaultExpSettings

The Maple procedure defaultExpSettings constructs the Maple record containing the default configuration settings for a polynomial-exponential system.

```
Settings := defaultExpSettings();
```

- `Settings`

  A Maple record containing the default configuration settings. The fields in this record are:

  - `algorithm`
  - `arithmeticType`
  - `precision`
  - `refineDigits`
  - `numRandomSystems`
  - `randomDigits`
  - `randomSeed`
  - `newtonOnly`
  - `numIterations`
  - `realityCheck`
  - `realityTest`
  - `deleteFiles`

Example

The following constructs the Maple record for the configurations settings described in Table 6.

```
> Settings := defaultExpSettings();
> Settings:-algorithm := 0;
> Settings:-arithmeticType := 1;
> Settings:-precision := 192;
```

6.4 defaultSettings

The Maple procedure defaultSettings constructs the Maple record containing the default configuration settings.

```
Settings := defaultSettings();
```

- `Settings`

  A Maple record containing the default configuration settings. The fields in this record are:

  - `algorithm`
  - `arithmeticType`
  - `precision`
  - `refineDigits`
  - `numRandomSystems`
  - `randomDigits`
  - `randomSeed`
  - `newtonOnly`
  - `numIterations`
  - `realityCheck`
  - `realityTest`
  - `deleteFiles`

Example

The following constructs the Maple record for the configurations settings described in Table 6.

```
> Settings := defaultSettings();
> Settings:-algorithm := 0;
> Settings:-arithmeticType := 1;
> Settings:-precision := 192;
```

6.5 loadOutput

The Maple procedure loadOutput loads output data.

PointsData := loadOutput(newtonOnly, algorithm, isReal, numVars);

- `newtonOnly`

  The value, either 0 or 1, of the configuration setting NEWTONONLY.

- `algorithm`

  The value, either 0, 1, or 2, of the configuration setting ALGORITHM.

- `isReal`

  A boolean value, i.e., either *true* or *false*, that describes if the input polynomial system is a real polynomial system.

- `numVars`

  The number of variables for the polynomial system.

- `PointsData`

  A Maple record containing output data from alphaCertified. See Section 6.1 for the structure of this record.

Example

```
> newtonOnly := 0;
> algorithm := 1;
> isReal := true;
> numVars := 2;
> PointsData := loadOutput(newtonOnly, algorithm, isReal, numVars);
```

6.6 printPoints

The Maple procedure printPoints constructs an input point file.

printPoints(pointsName, Points, arithmeticType);

- `pointsName`

  A string which is the name of the file to create.

- `Points`

  A two-dimensional list containing the points to print.

- `arithmeticType`

  The value, either 0 or 1, of the configuration setting ARITHMETICTYPE. If 0, the coordinates of the points are printed using a rational representation, otherwise, the points are printed using a floating point representation.

Example

```
> pointsName := "testPoints";
> Points := [[0,0],[1,1+I],[1,-I]];
> arithmeticType := 1;
> printPoints(pointsName, Points, arithmeticType);
```

6.7 printPolyExpSystem

The Maple procedure printPolyExpSystem constructs a polynomial-exponential system file.

printPolyExpSystem(funcName, Funcs, Vars);

- `funcName`

  A string which is the name of the file to create.

- `Funcs`

  A list containing the polynomial-exponential system.

- `Vars`

  A list containing the variables of the polynomial-exponential system.

Example

```
> funcName := "polyExpSystem";
> Funcs := [x * y^2 + x, y - exp(3 * x)];
> Vars := [x, y];
> printPolyExpSystem(funcName, Funcs, Vars);
```

6.8 printPolynomialSystem

The Maple procedure printPolynomialSystem constructs a polynomial system file.

printPolynomialSystem(polyName, Polys, Vars);

- `polyName`

  A string which is the name of the file to create.

- `Polys`

  A list containing the polynomial system.

- `Vars`

  A list containing the variables of the polynomial system.

Example

```
> polyName := "polynomialSystem";
> Polys := [x * y^2 + x, x * y - y];
> Vars := [x, y];
> printPolynomialSystem(polyName, Polys, Vars);
```

6.9 printSettings

The Maple procedure printSettings constructs a configuration settings file.

```
printSettings(settingsName, Settings);
```

- `settingsName`

  A string which is the name of the file to create.

- `Settings`

  A Maple record containing the configuration settings. See Section 6.4 for more details constructing this record.

Example

The following constructs the configurations settings file described in Table 6.

```
> settingsName := "settings";
> Settings := defaultSettings();
> Settings:-algorithm := 0;
> Settings:-arithmeticType := 1;
> Settings:-precision := 192;
> printSettings(settingsName, Settings);
```

Bibliography

[1] D.J. Bates, J.D. Hauenstein, A.J. Sommese, and C.W. Wampler. Bertini: Software for numerical algebraic geometry. Available at `www.nd.edu/~sommese/bertini`.

[2] L. Fousse, G. Hanrot, V. Lefèvre, P. Pélissier, and P. Zimmermann. MPFR: A Multiple-Precision Binary Floating-Point Library with Correct Rounding. *ACM Trans. Math. Softw.*, 33(2), Art. 13, 2007.

[3] T. Granlund. GNU MP: the GNU multiple precision arithmetic library. Available at `www.gmplib.org`.

[4] J.D. Hauenstein and F. Sottile. alphaCertified: certifying solutions to polynomial systems. To appear in *ACM Trans. Math. Softw.*

[5] J.D. Hauenstein and V. Levandovskyy. Certifying solutions to square systems of polynomial-exponential equations. Preprint, 2011. Available at `www.math.tamu.edu/~jhauenst/preprints`.

Appendix A

Configurations

The configurations for alphaCertified are presented below along with a brief description.

Table A.1: Configurations for alphaCertified

NAME	ACCEPTABLE VALUES	DEFAULT VALUE
ALGORITHM	0, 1, 2	2
ARITHMETICTYPE	0 or 1	0
PRECISION	$\geq$64	96
REFINEDIGITS	$\geq$0	0
NUMRANDOMSYSTEMS	$\geq$2	2
RANDOMDIGITS	> 0	10
RANDOMSEED	$>$0	random
NEWTONONLY	0 or 1	0
NUMITERATIONS	> 0	2
REALITYCHECK	-1, 0, or 1	1
REALITYTEST	0 or 1	0

Configurations

- ALGORITHM

  If ALGORITHM is 0, alphaCertified only determines which points are certifiably approximate solutions for the given polynomial system. If ALGORITHM is 1, alphaCertified also determines which certifiable approximate solutions correspond to distinct solutions. If ALGORITHM is 2 and the polynomial system is real, i.e., has only real coefficients, alphaCertified also determines which certifiable approximate solutions correspond to real solutions.

- ARITHMETICTYPE

  If ARITHMETICTYPE is 0, alphaCertified performs all computations using rational (certifiable) arithmetic. If ARITHMETICTYPE is 1, alphaCertified performs all computations using floating point arithmetic. In this case, the results of alphaCertified are *soft certified* since the floating point errors are not fully controlled. One way to control local errors is to increase PRECISION.

- PRECISION

  If ARITHMETICTYPE is 1, PRECISION indicates the starting level of precision (in bits). That is, all computations for each point start with this precision, but the internal working precision can be increased as needed. Standard settings include 64 bits (roughly 19 decimal digits), 96 (28), 128 (38), 160 (48), 192 (57), 224 (67), and 256 (77). In general, N bits is equivalent to $\lfloor N \log_{10}(2) \rfloor$ decimal digits.

- REFINEDIGITS

  If REFINEDIGITS is positive, say τ, all of the certifiable approximate solutions will be refined using Newton's method to be within $10^{-\tau}$ of the corresponding solution. If ARITHMETICTYPE is 1, the precision will automatically be increased internally so that the refined point is computed using a precision that has at least τ decimal digits.

- NUMRANDOMSYSTEMS

  When the polynomial system is overdetermined, alphaCertified will analyze NUMRANDOMSYSTEMS number of randomized square systems.

- RANDOMDIGITS

  When the polynomial system is overdetermined, each randomized square system must have a solution within $10^{-\tau}$ to be considered an approximate solution for the overdetermined system, where τ is RANDOMDIGITS.

- RANDOMSEED

  RANDOMSEED is the seed for the random number generator.

- NEWTONONLY

  If NEWTONONLY is 1, alphaCertified performs NUMITERATIONS number of Newton iterations on the points. If ARITHMETICTYPE is 1, the precision will automatically be increased internally following each iteration.

- NUMITERATIONS

  The number of Newton iterations to perform when NEWTONONLY is 1, that is, when alphaCertified is only setup to only perform Newton iterations on the points.

- REALITYCHECK

  If REALITYCHECK is -1, alphaCertified assumes that the polynomial system defines a real map. Otherwise, the value of REALITYCHECK instructs alphaCertified on which tests to perform to determine if the polynomial system defines a real map If REALITYCHECK is 0, alphaCertified only checks to see if all of the coefficients are real. If REALITYCHECK is 1, alphaCertified checks the coefficients as well as checking to see if the polynomial system, as a set, is invariant under conjugation.

- REALITYTEST

  If REALITYTEST is 0, the local approach for determining reality of associated solutions presented in [4] is used. If REALITYTEST is 1, the global approach presented in [4] is used.