Approximating all isolated solutions to polynomial systems using homotopy continuation methods – exercise session

Jan Verschelde

RAAG Summer School, Rennes, France, 30 June - 5 July, 2003

1. This first exercise is a test to see if phc runs on your system

   Create a file "input" with the following content:

   ```
   2
     x**2 + y**2 - 1;
     y - x**2;
   ```

   The input file starts with the number of equations and (optionally, but necessary in case of an unequal number) the number of unknowns. Observe that the $= 0$ is replaced by a semicolon. The exponentiation may also be denoted by a hat instead of a double star. Symbols that are forbidden to denote names of variables are i and I, because they both represent $\sqrt{-1}$. Also forbidden are e and E because they are used in the scientific notation of floating-point numbers, like $0.01 = 1.0e-2 = 1.0E-2$.

   Use the blackbox solver typing

   ```
   phc -b input output
   ```

   at the command prompt. Eventually, instead of just `phc` we have to type the complete path name for the location of the executable. The output of the solver will be sent to the file "output". In case the input file did not yet contain any solutions, the solution list will be appended to the input file.

2. The polynomial system of Bertrand Haas (which provided a counterexample for the conjecture of Koushnirenko) is represented as follows

   ```
   2
     x**108 + 1.1*y**54 - 1.1*y;
     y**108 + 1.1*x**54 - 1.1*x;
   ```

   For future use, save this system in the file `haas`. The best way to solve this system is via the total degree homotopy.

   If we just type in `phc` without any option, we run the program in full mode and will pass through all the main menus. We type in `haas` when the program asks us for the name of the input file. As the output may be rather large, we better save the output file on /tmp (if we are not on unix-like systems, we do not care). As we run through all the menus, for this system, a good choice is given by the default, so we can type in 0 to answer every question. Only at the very end, for the output format, it may be good to type in 1 instead of 0, so we can see the progress of the program as it adds solution after solution to the output file.

   The conjecture of Koushnirenko states that for a system like this, there can be at most four real solutions with all components different from zero. How many real solutions with all components different from zero does this system have? How many complex solutions?

3. Solve the following system (from MTH 191 organized by Bernd Sturmfels):

```
3
 x^3+y^2+z^2-1;
 x^2+y^3+z^2-1;
 x^2+y^2+z^3-1;
```

   How many solutions are there? How many are real?

   Do you observe the clustered solutions at the end? Also observe the estimate for the inverse of the condition number (the number after "rco" in the output). Solutions with high condition number close to each other indicate multiple roots.

   To increase our confidence in the numerical output we can apply Newton's method using multi-precision arithmetic. Call the program as `phc -v` (the option -v stands for validation) and select the 3rd option from the menu. As we increase the number of decimal places, we should not forget to decrease the tolerances on the error and residuals. For multiple roots we should allow more iterations, say 10, instead of the default number.

4. Consider the following system

```
   5
    (a-1)*(b-5)*(c-9)*(d-13) - 21;
    (a-2)*(b-6)*(c-10)*(f-17) - 22;
    (a-3)*(b-7)*(d-14)*(f-18) - 23;
    (a-4)*(c-11)*(d-15)*(f-19) - 24;
    (b-8)*(c-12)*(d-16)*(f-20) - 25;
```

   which is a valid input file for phc. Note that we replaced the logical `e` variable by `f`. A good name for this system is `multilin` (this is an example from Bernd Sturmfels).

   (a) Use the blackbox solver to solve this system. Compare the number of solutions to the total degree. How many real solutions does this system have?

   (b) Call `phc` in toolbox mode, as `phc -r` to explore various root counts. In particular, let phc compute all partitions of the set of unknowns to find the minimal multihomogeneous Bézout number.
   Before you leave `phc -r`, make sure to create a start system corresponding to this minimal multihomogeneous Bézout number. Save this system as `multilin_start`.

   (c) The path trackers in `phc` can be invoked directly via the option `phc -p`. Call `phc -p` with the input the system `multilin` and when prompted for a start system, type in `multilin_start`.
   Compare the output of `phc -p` to the output of the blackbox solver. In particular, do we see the same number of solutions? Do we see the same number of real solutions? Is there a difference in execution times?

5. The following example comes from enumerative geometry (and was communicated to me by Frank Sottile). In projective 3-space, there are 27 lines on a cubic surface. The cubic equation we consider has the equation

```
f(x0,x1,x2,x3) =
+(-8.38799345472072E-01 + 5.44440683670521E-01*i)*x0^3
+(-9.99708156627105E-01 + 2.41578470323001E-02*i)*x1^3
+(-8.04462206906772E-01 + 5.94003836400647E-01*i)*x2^3
+(-9.91423991980167E-01 + 1.30684613195698E-01*i)*x3^3 = 0
```

and the line is spanned by `[1,0,a,b]` and `[0,1,c,d]`. So we have four unknowns: `a,b,c` and `d`. The equations of the system arise from requiring that any linear combination of the two points satisfies the equation. The system we will solve is (in the input format to phc):

```
4
 +(-8.38799345472072E-01 + 5.44440683670521E-01*i)
 +(-8.04462206906772E-01 + 5.94003836400647E-01*i)*a**3
 +(-9.91423991980167E-01 + 1.30684613195698E-01*i)*b**3;
 +(-9.99708156627105E-01 + 2.41578470323001E-02*i)
 +(-8.04462206906772E-01 + 5.94003836400647E-01*i)*c**3
 +(-9.91423991980167E-01 + 1.30684613195698E-01*i)*d**3;
 b**2*d + a**2*c;
 b*d**2 + a*c**2;
```

(a) Use the blackbox solver of phc to solve this system.
Observe the root counts and in particular the differences between total degree, product bounds, and mixed volume. Do you find 27 lines?

(b) Instead of the arbitrary complex coefficients of the cubic surface, let us take nice floating-point numbers:

```
f(x0,x1,x2,x3) =
- 5.44440683670521*x0^3 - 9.99708156627105*x1^3
+ 0.804462206906772E-01*x2^3 + 2.30684613195698E-01*x3^3 = 0
```

and solve the system

```
4
 -5.44440683670521 + 0.804462206906772E-01*a**3
                   + 2.30684613195698E-01*b**3;
 -9.99708156627105 + 0.804462206906772E-01*c**3
                   + 2.30684613195698E-01*d**3;
 b**2*d + a**2*c;
 b*d**2 + a*c**2;
```

This system is small enough to solve from scratch, but morally (and for the sake of this exercise) a secant homotopy must be used between the system we first solved and the new system with the nice floating-point coefficients. So use `phc -p`.
How many of the 27 lines are real?