B-378

CMPSc : A **C**ontinuation **M**ethod for **P**olynomial **S**ystems (C++ version)

Sunyoung Kim[†] and Masakazu Kojima[‡], March 2002

Abstract. **CMPS**c is a C++ program for homotopy continuation methods to find all isolated solutions of a system of polynomial equations. The numerical methods for solving polynomial systems employed in **CMPS**c are equivalent to the ones used in a matlab program **CMPS**m. The focus of **CMPS**c is on implementation of the cheater's homotopy. The software CMPSc, this manual and some numerical examples are available at http://www.is.titech.ac.jp/∼kojima/ CMPSc/

Key words. Systems of Polynomials, Homotopy Continuation Methods, Polyhedral Homotopy, Cheater's Homotopy.

† Department of Mathematics, Ewha Women's University
11-1 Dahyun-dong, Sudaemoon-gu, Seoul 120-750 Korea
email: *skim@ewha.ac.kr, skim@is.titech.ac.jp*
A considerable amount of this work was conducted while this author was visiting Tokyo Institute of Technology, Dept. of Math. and Comp. Sci.. Research supported by Kosef R004-2001-00200.

‡ Department of Mathematical and Computing Sciences
Tokyo Institute of Technology
2-12-1 Oh-Okayama, Meguro-ku, Tokyo 152-8552 Japan
e-mail: *kojima@is.titech.ac.jp*

1 Introduction

We consider a system of polynomial equations $\boldsymbol{f}(\boldsymbol{x}) = \boldsymbol{0}$ from the n-dimensional complex space $\mathbb{C}^n$ into itself, where $\boldsymbol{f}(\boldsymbol{x}) = (f_1(\boldsymbol{x}), f_2(\boldsymbol{x}), \ldots, f_n(\boldsymbol{x})) \in \mathbb{C}^n$ denotes a system of polynomial in variables $x_1, x_2, \ldots, x_n \in \mathbb{C}$ and $\boldsymbol{x} = (x_1, x_2, \ldots, x_n)$ a variable vector in $\mathbb{C}^n$. As in [6] we consider the cyclic-3 polynomial [3] for an example throughout the paper:

$$1 - x_1x_2x_3 = 0,\ x_1x_2 + x_2x_3 + x_3x_1 = 0,\ x_1 + x_2 + x_3 = 0 \tag{1}$$

Homotopy continuation methods [1, 8, 9, 13, 14, 15, 16, etc.] are known as powerful numerical methods for computing all isolated solutions of $\boldsymbol{f}(\boldsymbol{x}) = \boldsymbol{0}$. A strategy behind the methods is to prepare a homotopy (polynomial) function $\boldsymbol{h} : \mathbb{C}^n \times [0, 1] \to \mathbb{C}^n$ such that

(a) all solutions of *the starting polynomial system* $\boldsymbol{h}(\boldsymbol{x}, 0) = \boldsymbol{0}$ are easily attainable,

(b) *the target polynomial system* $\boldsymbol{h}(\boldsymbol{x}, 1) = \boldsymbol{0}$ coincides with $\boldsymbol{f}(\boldsymbol{x}) = \boldsymbol{0}$.

Then, trace a solution curve of $\boldsymbol{h}(\boldsymbol{x}, t) = \boldsymbol{0}$ numerically in the space $\mathbb{C}^n \times [0, 1]$ starting from a known solution $\boldsymbol{x}^0$ of $\boldsymbol{h}(\boldsymbol{x}, 0) = \boldsymbol{0}$ with *the homotopy parameter* $t = 0$. Increasing the value of t leads to a solution of the target system $\boldsymbol{h}(\boldsymbol{x}, 1) \equiv \boldsymbol{f}(\boldsymbol{x}) = \boldsymbol{0}$ at $t = 1$. Linear, polyhedral and the cheater's homotopies represent most popular homotopies. CMPSc is a C++ program that handles the cheater's homotopy whereas CMPSm [6] is a matlab program that can be used for the three homotopies. Both implement a path-following method based on predictor-corrector procedures for tracing solution curves of homotopy systems. Some differences between CMPSc and CMPSm lie, however, in function $f(x)$ and its Jacobian evaluation. CMPSc utilizes many matlab-provided functions for complex arithmetic, ... , etc., while CMPSm handles those in our own C++ codes. We used routines from Numeical recipes in C [11] for linear system solvers including singular value decomposition. Some technical details of the method were presented in the recent paper [4].

2 Notations and Symbols

Let $\mathbb{R}$ and $\mathbb{Z}_+$ denote the set of real numbers and the set of nonnegative integers, respectively. For every $\boldsymbol{x} = (x_1, x_2, \ldots, x_n) \in \mathbb{C}^n$ and $\boldsymbol{a} = (a_1, a_2, \ldots, a_n) \in \mathbb{Z}^n_+$, we use the notation $\boldsymbol{x}^{\boldsymbol{a}}$ for the term $x_1^{a_1}x_2^{a_2}\cdots x_n^{a_n}$. Then we can write each $f_k(\boldsymbol{x})$ of $\boldsymbol{f}(\boldsymbol{x})$ as $f_k(\boldsymbol{x}) = \sum_{\boldsymbol{a}\in\mathcal{A}_k} c_k(\boldsymbol{a})\boldsymbol{x}^a$ for a finite subset $\mathcal{A}_k$ of $\mathbb{Z}^n_+$ and $c_k(\boldsymbol{a}) \in \mathbb{C}$ ($\boldsymbol{a} \in \mathcal{A}_k$) ($k = 1, 2, \ldots, n$). We call $\mathcal{A}_k$ the *support* of $f_k(\boldsymbol{x})$. For the cyclic-3 polynomial (1), we take

$$\left.\begin{array}{l}
\boldsymbol{a}_1^1 = (0,0,0),\ \boldsymbol{a}_1^2 = (1,1,1),\ \mathcal{A}_1 = \{\boldsymbol{a}_1^1, \boldsymbol{a}_1^2\}, \\
\boldsymbol{a}_2^1 = (1,1,0),\ \boldsymbol{a}_2^2 = (0,1,1),\ \boldsymbol{a}_2^3 = (1,0,1),\ \mathcal{A}_2 = \{\boldsymbol{a}_2^1, \boldsymbol{a}_2^2, \boldsymbol{a}_2^3\}, \\
\boldsymbol{a}_3^1 = (1,0,0),\ \boldsymbol{a}_3^2 = (0,1,0),\ \boldsymbol{a}_3^3 = (0,0,1),\ \mathcal{A}_3 = \{\boldsymbol{a}_3^1, \boldsymbol{a}_3^2, \boldsymbol{a}_3^3\}, \\
c_1(\boldsymbol{a}_1^1) = 1,\ c_1(\boldsymbol{a}_1^2) = -1,\ c_2(\boldsymbol{a}_2^1) = 1,\ c_2(\boldsymbol{a}_2^2) = 1,\ c_2(\boldsymbol{a}_2^3) = 1, \\
c_3(\boldsymbol{a}_3^1) = 1,\ c_3(\boldsymbol{a}_3^2) = 1,\ c_3(\boldsymbol{a}_3^3) = 1.
\end{array}\right\} \tag{2}$$

3 Cheater's homotopy

Since CMPSc employs a cheater's homotopy for tracing solution curves, we give brief descriptions of a cheater's homotopy. Each component $h_k(\boldsymbol{x}, t)$ of the homotopy whose zeros (*i.e.* homotopy solution curve) can be traced by CMPSc is written as

$$h_k(\boldsymbol{x}, t) = \sum_{\boldsymbol{a} \in \mathcal{A}_k} \left((1-t)\tilde{c}_k(\boldsymbol{a}) + t c_k(\boldsymbol{a})\right) \boldsymbol{x}^{\boldsymbol{a}} t^{\rho_k(\boldsymbol{a})}. \tag{3}$$

Here $\tilde{c}_k(\boldsymbol{a}) \in \mathbb{C}$ and $0 \leq \rho_k(\boldsymbol{a}) \in \mathbb{R}$ are given numbers ($\boldsymbol{a} \in \mathcal{A}_k$). To avoid possible degeneracy while tracing homotopy solution curves, a common practice is to assign randomly generated complex numbers to all (or some) of $\tilde{c}_k(\boldsymbol{a})$'s. Obviously $\boldsymbol{h}(\boldsymbol{x}, 1) = \boldsymbol{f}(\boldsymbol{x})$ for every $\boldsymbol{x} \in \mathbb{C}^n$; hence (b) holds. We call $\rho_k(\boldsymbol{a})$ ($\boldsymbol{a} \in \mathcal{A}_k,\ k = 1, 2, \ldots, n$) *powers* of t. Given solutions $\boldsymbol{x}^1, \boldsymbol{x}^2, \ldots, \boldsymbol{x}^\ell$ of $\boldsymbol{h}(\boldsymbol{x}, 0) = \boldsymbol{0}$, and the data $\mathcal{A}_k$, $c_k(\boldsymbol{a})$ ($\boldsymbol{a} \in \mathcal{A}_k$), $\tilde{c}_k(\boldsymbol{a})$ ($\boldsymbol{a} \in \mathcal{A}_k$) and $\rho_k(\boldsymbol{a})$ ($\boldsymbol{a} \in \mathcal{A}_k$) ($k = 1, 2, \ldots, n$), we can apply CMPSc to trace homotopy solution curves of $\boldsymbol{h}(\boldsymbol{x}, t) = \boldsymbol{0}$ numerically, starting from $\boldsymbol{x}^1, \boldsymbol{x}^2, \ldots, \boldsymbol{x}^\ell$ at $t = 0$. If $\rho_k(\boldsymbol{a}) = 0$ ($\boldsymbol{a} \in \mathcal{A}_k,\ k = 1, 2, \ldots, n$) in (3), the resulting homotopy is a linear homotopy.

The cheater's homotopy [9, etc.] is a combination of linear and polyhedral homotopies with randomly generated values $\tilde{c}_k(\boldsymbol{a})$ and $c_k(\boldsymbol{a})$ ($\boldsymbol{a} \in \mathcal{A}_k,\ k = 1, 2, \ldots, n$). To generate the cheater's homotopy, we produce multiple homotopy functions $\boldsymbol{h}^p$ with each homotopy system as

$$h_k^p(\boldsymbol{x}, t) = \sum_{\boldsymbol{a} \in \mathcal{A}_k} \left((1-t)\tilde{c}_k(\boldsymbol{a}) + t c_k(\boldsymbol{a})\right) \boldsymbol{x}^{\boldsymbol{a}} t^{\rho_k^p(\boldsymbol{a})} \tag{4}$$

($k = 1, \ldots, n$). Here $\tilde{c}_k(\boldsymbol{a}) \in \mathbb{C}$ ($\boldsymbol{a} \in \mathcal{A}_k,\ k = 1, 2, \ldots, n$) are randomly generated complex numbers and $\rho_k^p(\boldsymbol{a})$ ($\boldsymbol{a} \in \mathcal{A}_k,\ k = 1, 2, \ldots, n$) are nonnegative real numbers chosen according to the theory of the polyhedral homotopy continuation method [9]. The number of paths to be traced in the cheater's homotopy is half the number of paths in polyhedral homotopy. This generates computational advantages of the cheater's homotopy.

The distinctive feature of the cheater's homotopy system $\boldsymbol{h}^p(\boldsymbol{x}, t)$ is that exactly two of $\rho_k(\boldsymbol{a})$ ($\boldsymbol{a} \in \mathcal{A}_k$) are zeros and all the others are positive for each $k = 1, 2, \ldots, n$. As a result, the starting polynomial system $\boldsymbol{h}^p(\boldsymbol{x}, 0) = \boldsymbol{0}$ is a binomial system whose solutions can be easily found. CMPSc can start from the solutions of a binomial system with the coefficient $\tilde{c}_k(\boldsymbol{a})$'s at $t = 0$ and attain solutions of the target polynomial system at $t = 1$ by tracing homotopy solution curves. When applying CMPSc, we need to scale the homotopy parameter t so that all positive $\rho_k(\boldsymbol{a})$'s are not less than 1. Such a scaling is always possible. In fact, replace the homotopy parameter t by $t = s/\rho^*$ where ρ^* denotes the minimum of all positive $\rho_k(\boldsymbol{a})$'s ($\boldsymbol{a} \in \mathcal{A}_k,\ k = 1, 2, \ldots, n$). Then the modified homotopy $\hat{\boldsymbol{h}}(\boldsymbol{x}, s) = \boldsymbol{h}(\boldsymbol{x}, s/\rho^*)$ satisfies the requirements (a) and (b) in Section 1.

All data to implement the cheater's homotopy for the cyclic-3 polynomial (1) are stored in the directory 3cycC. Specifically, the coefficient file 3cycC/3cyc.coef includes the data $c_k(\boldsymbol{a})$ and $\tilde{c}_k(\boldsymbol{a})$ ($\boldsymbol{a} \in \mathcal{A}_k,\ k = 1, 2, 3$) shown as follows.

```
# 3cycC.coef
# Cheater's Homotopy
```

```
n = 3
m = 2 3 3
a1.1 = 1 1 1
a1.2 = 0 0 0
a2.1 = 1 1 0
   ...
a3.3 = 0 0 1
coef1.1r = -1
coef1.1i = 0
   ...
coef3.3i = 0
Rcoef1.1r = 0.32103329826836
   ...
Rcoef3.3i = -0.426542587548004
```

This file contains the six keywords "n = ", "m = ", "a$k.\ell$ = ", "coef$k.\ell$r = ", "coef$k.\ell$i = ", "Rcoef$k.\ell$r = ", and "Rcoef$k.\ell$i = ", where k and ℓ stand for positive integers. Lines starting with '♯' are regarded as comments by CMPSc. Following two lines of comments in the beginning, the dimension 3 of the problem is specified by "n = 3", and then the cardinalities of $\mathcal{A}_1$, $\mathcal{A}_2$, $\mathcal{A}_3$ by "m = 2 3 3". They are followed by the description of the supports $\mathcal{A}_1$, $\mathcal{A}_2$, $\mathcal{A}_3$ of the cyclic-3 polynomial where each line denotes a$k.\ell$ $= \boldsymbol{a}_k^\ell \in \mathcal{A}_k$ $(k = 1, 2, 3)$. Then each of the next several lines denotes

$$\text{coef}k.\ell\text{r} \;=\; \text{the real part of } c_k(\boldsymbol{a}_k^\ell) \;\text{ or coef}k.\ell\text{i} \;=\; \text{the imaginary part of } c_k(\boldsymbol{a}_k^\ell),$$

and each line of the last part means

$$\text{Rcoef}k.\ell\text{r} \;=\; \text{the real part of } \tilde{c}_k(\boldsymbol{a}_k^\ell) \;\text{ or Rcoef}k.\ell\text{i} \;=\; \text{the imaginary part of } \tilde{c}_k(\boldsymbol{a}_k^\ell),$$

respectively.

The cell files of initial points 3cycC/3cycC1 and 3cycC/3cycC2 contain powers of t and the initial solutions of the binomial systems. The solutions of the starting polynomial system $\boldsymbol{h}^1(\boldsymbol{x}, 0) = \boldsymbol{0}$ are described in a *cell file of initial points* 3cycC/3cycC1:

```
# 3cycC1
# Cheater's homotopy
NumOfSols = 3
# Root information
sol1 = 0.936739515003483
-0.699702958948314
0.0993589571194037
0.607291676305282
-1.0026405118961
-1.12951867075905
```

```
   ...
sol3 = -1.07433029505412
   ...
# Power information
=  0   0
=  0   0   4.75143
=  0   0   1.9309
```

This file includes the three key words "NumOfSols = ", "sol$k$ = ", and "= ", where k stands for positive integers. Any line starting with '♯' is neglected by CMPSc. As specified by NumOfSols = 3, this file contains 3 solutions, sol1, sol2, sol3 of the starting polynomial system $\boldsymbol{h}(\boldsymbol{x}, 0) = \mathbf{0}$. The kth solution, sol$k = (x_1, x_2, x_3)$ is described such that

$$\begin{array}{l} \text{sol}k = \text{the real part of } x_1 \\ \text{the imaginary part of } x_1 \\ \ldots \\ \text{the real part of } x_3 \\ \text{the imaginary part of } x_3. \end{array}$$

Each line of the last part of the file provides the powers $\rho_k(\boldsymbol{a})$ ($\boldsymbol{a} \in \mathcal{A}_k$) of the homotopy parameter t ($k = 1, 2, 3$).

In the case of the cyclic-3 polynomial (1), we constructed two polyhedral homotopies based on the polyhedral homotopy theory; the one has the set of powers

$$\left.\begin{array}{l} \rho_1(\boldsymbol{a}_1^1) = \rho_1(\boldsymbol{a}_1^2) = 0,\ \rho_2(\boldsymbol{a}_2^1) = \rho_2(\boldsymbol{a}_2^2) = 0,\ \rho_2(\boldsymbol{a}_2^3) = 4.75143, \\ \rho_3(\boldsymbol{a}_3^1) = \rho_3(\boldsymbol{a}_3^2) = 0,\ \rho_3(\boldsymbol{a}_3^3) = 1.9309 \end{array}\right\} \tag{5}$$

stored with 3 starting solutions in a cell file of initial points 3cycC/3cycC1, and the other has the set of powers

$$\left.\begin{array}{l} \rho_1(\boldsymbol{a}_1^1) = \rho_1(\boldsymbol{a}_1^2) = 0,\ \rho_2(\boldsymbol{a}_2^1) = 1.9309,\ \rho_2(\boldsymbol{a}_2^2) = \rho_2(\boldsymbol{a}_2^3) = 0, \\ \rho_3(\boldsymbol{a}_3^1) = 0,\ \rho_3(\boldsymbol{a}_3^2) = 4.75143,\ \rho_3(\boldsymbol{a}_3^3) = 0 \end{array}\right\} \tag{6}$$

together with 3 starting solutions in another cell file of initial points 3cycC/3cycC2. All other data $\mathcal{A}_k$ ($k = 1, 2, 3$), $c_k(\boldsymbol{a}) = \tilde{c}_k(\boldsymbol{a})$ ($\boldsymbol{a} \in \mathcal{A}_k,\ k = 1, 2, 3$) and $\tilde{c}_k(\boldsymbol{a})$ ($\boldsymbol{a} \in \mathcal{A}_k,\ k = 1, 2, 3$) are located in the coefficient file 3cycC/3cycC.coef.

```
# 3cycC1
# Power information
=  0   0
=  0   0   4.75143
=  0   0   1.9309

# 3cycC2
# Power information
=  0   0
=  1.9309   0   0
=  0   4.75143   0
```

4 Parameter file

Executing CMPSc requires three input files. One of them is a parameter file named para3cycC.dat. As we can see from the content of the following example for executing CMPSc to solve the cyclic-3 polynomial by the cheater's homotopy described in Section 2.1, a user can specify the parameters accINfVal, accInNewtonDir, beta, divMagOFx, dTauMax, minEigForNonsing, NewtonDirMax and predItMax. The change of values in these parameters may result in shorter or longer cpu time and different approximate solutions of the given polynomial system. The parameter file provides tools to control accuracy of solutions and overall performance of CMPSc by specifying various values for the parameters.

```
# para3cycC.dat
# default accINfVal= 1.0e-10
accINfVal= 1.e-10
# default accInNewtonDir= 1.e-8
accInNewtonDir= 1.e-8
# default beta=5, 30, 50, 100, 200, 1000 for cyc8, 9, 10, 11, 12 and 13
beta=1;
# divMagOFx = 1.0e+3 for cyclic problems
divMagOFx= 1.0e+3
# default dTauMax= 0.1
dTauMax= 0.1
# default minEigForNonsing= 1.0e-12
minEigForNonsing= 1.e-12
# default value NewtonDirMax= 0.1
NewtonDirMax= 0.1
# default predItMax= 2000
predItMax= 2000
```

Here "accINfVal=", "accInNewtonDir=", "beta=", "divMagOFx=", "dTauMax=", "minEigForNonsing=", "NewtonDirMax=", "predItMax=" are the names of the parameters at the first column of a line, and lines beginning with '♯' are regarded as comments by CMPSc. The role of the parameters is to present criteria for stopping execution of CMPSc, control predictor step lengths, and lower the highest power of the continuation parameter t so that it can contribute numerically easier path following.

The values of "accINfVal", "accInNewtonDir" and "minEigForNonsing" are used to determine an approximate solution $\boldsymbol{x}$ nonsingular or singular based on the following two tests.

$$\left\{\begin{array}{l}\max\{|f_k(\boldsymbol{x})| : k = 1, 2, \ldots\} \leq \text{accINfVal},\\ \|\boldsymbol{Df}(\boldsymbol{x})^{-1}\boldsymbol{f}(\boldsymbol{x})\| \leq \text{accInNewtonDir, and}\\ \text{the minimum eigenvalue of } \boldsymbol{Df}(\boldsymbol{x})^*\boldsymbol{Df}(\boldsymbol{x}) \geq \text{minEigForNonsing}\end{array}\right.$$

or

$$\left\{\begin{array}{l}\max\{|f_k(\boldsymbol{x})| : k = 1, 2, \ldots, n\} \leq \text{accINfVal, and}\\ \text{the minimum eigenvalue of } \boldsymbol{Df}(\boldsymbol{x})^*\boldsymbol{Df}(\boldsymbol{x}) < \text{minEigForNonsing.}\end{array}\right.$$

If an approximate solution $\boldsymbol{x}$ passes the first test, then the solution is regarded as a nonsingular solution. If an approximate solution $\boldsymbol{x}$ fails the first test but passes the second test, then it is determined to be a singular solution.

Divergent solutions are detected using "divMagOFx": when the 2-norm of an iterate $\boldsymbol{x}$ becomes larger than the value of divMagOFx, CMPSc stops following a homotopy path. In the case that CMPSc consumes more predictor iterations than predItMax, it stops processing further.

A user can control the largest step size with the parameter "dTauMax" because it provides an upper bound for increases in $t^{\rho_k(\boldsymbol{a})}$ $(\boldsymbol{a} \in \mathcal{A}_k, k = 1, 2, \ldots, n)$ in the predictor. When the 2-norm $\|d\boldsymbol{x}\|$ of the Newton direction is greater than the value NewtonDirMax$\times\|\boldsymbol{x}\|$, the corrector iteration is discarded and a predictor iteration with a smaller step is retried. The smaller the values of these two parameters are, the more accurate CMPSc traces a homotopy curve. This may prevent a jump from a curve into another curve in the middle of path following.

The parameter beta ≥ 1 serves as a scaling for the homotopy parameter t. For the cheater's homotopy, we suggest to take

$$1 \leq \text{ beta } \approx \gamma \times \text{ "the maximum of powers, } \rho_k(\boldsymbol{a})\text{'s over all homotopies",}$$
$$0.01 \leq \gamma \leq 0.1$$

for computational efficiency and numerical stability. When large dimensional problems are solved by the cheater's homotopy, the maximum of $\rho_k(\boldsymbol{a})$'s over all homotopies can be quite large, for example, it can exceed 200,000. Taking an appropriate beta significantly improves the performance of CMPSc. When the maximum is small, for example, less than 100, choose beta $= 1$.

Remark. Three additional parameters "informationLevel", "coeffSW" and "modSW" are set in the routine inputQ.cpp. After changing the parameters in the routine inputQ.cpp, recompilation and link by "make" using makefile is necessary to have updated parameters.

5 Execution of CMPSc

We briefly explain the procedure to solve the cyclic-3 polynomial using the cheater's homotopy (4) with (2): For solutions of the cyclic-3 problem by the cheater's homotopy with the data files 3cycC.coef, 3cycC1, 3cycC2 and a parameter file para3cycC.dat. When we use the cheater's homotopy, we need to specify several "cellFileInitPt"s for input arguments in addition to a parameter file and a coefficient file. CMPSc also accepts input arguments, "startCellNo" and "endCellNo". "makefile" is provided with source codes. CMPSc has been compiled with gnu g++ compiler version 2.96 on linux system and version 2.95.2 on Sun OS system. The following is a sequence of commands to execute the program.

(i) Compile and link the routines using make.

```
> make
```

(ii) CMPSc needs the following input files in a directory.

- para3cycC.dat — a parameter file,
- 3cycC.coef, 3cycC1 3cycC2 — a coefficient file and cell files of initial points.

The location of the files can be in the same directory as CMPSc or a different directory. In the latter case, a path to the directory should be given at the time of execution.

(iii) Execute CMPSc as

```
> CMPSc para3cycC.dat 3cycC.coef 1 2
```

if the input files and cell files 3cycC1 and 3cycC2 exist in the same directory. Or,

```
> CMPSc ~/home/3cycC/para3cycC.dat ~/home/3cycC/3cycC.coef 1 2
```

if the input files and two cell file are in the directory ∼/home/3cycC. Here we take startCellNo= 1 and endCellNo= 2.

6 Output files

The information on statistics of path following and solutions are main output of CMPSc. The names of output files are composed of prefix “3cycC”, startCellNo (*e.g.* 1), endCellNo (*e.g.* 2), and suffix “.stat” for statistics or suffix “.sol” for solution information, respectively. startCellNo and endCellNo are given at the time of running CMPSc as shown in the previous section.

“informationLevel” in the routine inputQ.cpp is used to have more or less information print on the screen and in files. For instance, if a user wants to print only limited statistics and solutions in the two files mentioned above, “informationLevel” needs to be set to −2. A value greater than and equal to 0 for informationLevel produces an additional output file called with the suffix “.note”. The preffix is the same as the files with statistics and solutions. More information is available inside the routine inputQ.cpp.

The following is an example of the *.stat file from the cyclic-9 polynomial stored in the directory 9cycC. It is a result of executing

```
> CMPSc para9cycC.dat 9cycC.coef 1 978
```

in the directory 9cycC.

```
# 9cycC1-978.stat
#  cell  init statusP  pIT  TcIT   cpu hValError     normOFx      minEig

   ...

     49     1  +3        65   135   4.4  4.18e-15  6.8541e+00  +6.9948e-01
```

```
  ...

    49    10  +4        69   224   4.6  2.13e-12  2.6180e+00  +3.6930e-13

  ...

    50     1  -2       217   993  19.4  1.02e-01  7.3932e+01  +2.8480e-16
```

Here "cell" denotes a positive integer attached to a cell file of initial points, which takes a value from "startCellNo" to "endCellNo" (the last two input arguments). "init" means the initial solution number. "statusP" indicates whether the homotopy path converged to a nonsingular solution (statusP = 3), converged to a singular solutions (statusP = 4) or diverges (statusP = -2). Also statusP can have other values in different circumstances. The meaning of the values is described at the top of the *.stat file. "pIT", "TcIT" and "cpu " stand for the total number of predictor iterations, the total number of corrector iterations and the cpu time spent to trace the homotopy path, respectively. "hValError", "normOFx" and "minEig" denote the 1-norm of errors in function values of an approximate solution $\boldsymbol{x}$ computed, the 2-norm of $\boldsymbol{x}$ and the minimum eigenvalue of $\boldsymbol{Df}(\boldsymbol{x})^{*}\boldsymbol{Df}(\boldsymbol{x})$, respectively. These values are meaningful only when statusP is either +3 or +4.

The other output file is a solution file. For the cyclic-9 polynomial problem, we have

```
# 9cycC1-978.sol
+7.6604444311e-01 . . . +9.3781448372e-02  +4.1789657759e-15  49  1


  ...


+1.7364863502e-01 . . . -3.7616321836e-01  +2.1321577253e-12  49 10
```

Each line without comment mark # shows an approximate solution $\boldsymbol{x}$, the 1-norm of errors in function values, the cell number of initial points and the initial solution number shown as

$$\mathrm{real}(x_1), \mathrm{imag}(x_1), \ldots, \mathrm{real}(x_n), \mathrm{imag}(x_n), \mathrm{hValError}, \mathrm{cell}, \mathrm{init},$$

where the last three numbers correspond to "hValError', "cell" and "init" in the *.stat file.

7 Numerical results

Table 7 shows numerical results on the economic polynomial with dimensions 6, 7, ..., 10 solved by cheater's homotopy with the data stored in the directories 6ecoC, 7ecoC, ..., 10ecoC. The computation was done on Pentium III 1.8GHz CPU, the same machine that the numerical results from CMPSm [6] were obtained. Here the following notation is used:

Table 1: Numerical results from economic n problems by cheater's homotopy

n	6	7	8	9	10
No.paths	16	32	64	128	256
Av.pred.it.	105.1	94.2	97.7	107.7	137.6
Av.corr.it.	210.0	183.9	203.5	228.4	310.6
Av.CPU	0.3	0.4	0.5	0.9	1.5
No.sol.	16	32	64	128	256

n	: the number of variables.
No.paths	: the number of paths traced
Av.pred.it.	: the average number of predictor iterations per path.
Av.corr.it.	: the average number of corrector iterations per path.
Av.CPU	: the average CPU time per path.
No.sol.	: the number of (nonsingular) solutions computed.

Av.CPU in Table 7 shows that CMPSc is faster than CMPSm to solve the same problems. The average numbers of predictor and corrector iterations are almost equivalent to those of CMPSm.

8 Concluding remarks

CMPSc may not be robust to find all solutions of a polynomial system for some cases, although the failure rate to find a solution is very low, less than 0.1% from our experience. Two effective techniques exist to recover missing solutions. The one is to recompute the homotopy curves that have converged to a solution with more conservative parameters; take dTauMax= 0.01 and NewtonDirMax= 0.01 instead of the default values 0.1 and 0.1. The other is to apply CMPSc to more than a set of homotopy functions for a polynomial system to be solved, and merge multiple sets of the solutions obtained into a set of the solutions; choosing a different $\beta \geq 1$ in the parameter file and/or reseting coeffSW in the routine inputQ.cpp to -1 in CMPSc would yield different homotopies. See [4] for more details.

An important advantage of homotopy continuation methods lies in parallel computation. Indeed, we can trace all homotopy paths simultaneously in parallel if the input data are split in a consistent manner. CMPSc has been designed to benefit from this feature. For example, if three different cpus are available, we can execute each of the following commands

```
>CMPSc  para9cycC.dat 9cycC.coef 9cycC 1 300
>CMPSc  para9cycC.dat 9cycC.coef 9cycC 301 600
>CMPSc  para9cycC.dat 9cycC.coef 9cycC 601 978
```

in one of three cpus, instead of issuing a command

```
>CMPSc para9cycC.dat 9cycC.coef 9cycC 1 978
```

in a single cpu to solve the economic problem of the dimension 9 with the data stored in 9cycC.

CMPSc has been developed as a part of a joint project [4, 12] of (parallel) implementation of polyhedral homotopy continuation methods with Yang Dai, Katsuki Fujisawa, Sunyoung Kim, Masakazu Kojima and Akiko Takeda. This C++ version CMPSc implements the same numerical methods to its Matlab version CMPSm for path following. A partial outcome of this project was reported in [4] with some numerical results on the cyclic polynomial system of the dimensions $8, 9, \ldots, 12$, which was obtained by a C++ code implementing the cheater's homotopy continuation method. More recently, they have succeeded in approximating all isolated solutions of the cyclic-13 polynomial system by a parallel implementation of the C++ code. It resulted in 2,704,156 isolated solutions (counting multiplicity). See [7].

Acknowledgment. Most of input data for cyclic and economic polynomials in this paper and the homepage http://www.is.titech.ac.jp/∼kojima/CMPSc was constructed using Professor Dai's and Dr. Takeda's C++ programs. The authors would like to express appreciation for their help.

References

[1] E. Allgower and K. Georg, *Numerical continuation methods,* Springer-Verlag, 1990.

[2] D. N. Bernshtein, "The number of roots of a system of equations," *Functional Analysis and Appl.* **9**(3) (1975) 183–185.

[3] G. Björck and R. Fröberg, "A faster way to count the solutions of inhomogeneous systems of algebraic equations, with applications to cyclic n-roots", *Journal Symbolic Computation* **12** (1991) 329-336.

[4] Y. Dai, S. Kim and M. Kojima, "Computing all nonsingular solutions of cyclic-n polynomial using polyhedral homotopy continuation methods," B-373, Dept. of Math. and Comp. Sciences, Tokyo Inst. of Tech., September 2001.

[5] B. Huber and B. Sturmfels, "A Polyhedral method for solving sparse polynomial systems," *Mathematics of Computation* **64** (1995) 1541–1555.

[6] S. Kim and M. Kojima, "**CMPS**m: A **C**ontinuation **M**ethod for **P**olynomial **S**ystems (MATLAB version)," B-376, Dept. of Math. and Comp. Sciences, Tokyo Inst. of Tech., January 2002.

[7] M. Kojima, his web site: "http://www.is.titech.ac.jp/∼kojima/polynomials/index.html."

[8] T. Y. Li, "Solving polynomial systems," *The mathematical intelligencer*, **9**, 3 (1987) 33-39.

[9] T. Y. Li, "Solving polynomial systems by polyhedral homotopies", *Taiwan Journal of Mathematics* **3** (1999) 251-279.

[10] T. Y. Li and X. Li, "Finding Mixed Cells in the Mixed Volume Computation," *Foundation of Computational Mathematics* **1** (2001) 161-181.

[11] W. H. Press, S. A. Teukolsky, W. T. Vetterling, and B. P. Flannery, *Numerical Recipes in C: the art of scientific computing*, 2nd edition, Cambridge University Press, 1992.

[12] A. Takeda, M. Kojima, and K. Fujisawa, "Enumeration of all solutions of a combinatorial linear inequality system arising from the polyhedral homotopy continuation method," to appear in *J. of Operations Society of Japan.*

[13] J. Verschelde, The database of polynomial systems is in his web site: "http://www.math.uic.edu/∼jan/".

[14] J. Verschelde, P. Verlinden and R. Cools, "Homotopies exploiting Newton polytopes for solving sparse polynomial systems," *SIAM J. Numerical Analysis*, **31** (1994) 915-930.

[15] J. Verschelde, "Algorithm 795: PHCpack: A general-purpose solver for polynomial systems by homotopy continuation," *ACM Trans. Math. Softw.* **25** (1999) 251-276.

[16] L. T. Watson, M. Sosonkina, R. C. Melville, A. P. Morgan, and H. F. Walker, "HOMPACK90: A suite of Fortran 90 codes for globally homotopy algorithms," *ACM Trans. Math. Softw.* **23**, 4 (1997) 514-549.