PHCpack: A Software Platform for Numerical Algebraic Geometry

Jan Verschelde

Department of Math, Stat & CS
University of Illinois at Chicago
Chicago, IL 60607-7045, USA

email: `jan@math.uic.edu`

URL: `http://www.math.uic.edu/~jan`

IMA Workshop Software for Algebraic Geometry

23-27 October 2006, Minneapolis.

PHC = Polynomial Homotopy Continuation

- `phc -b` released PHCpack 1.0 in August 1997
  Algorithm 795 ACM TOMS
- experimental platform to validate the algorithms described in [Andrew J. Sommese & Charles W. Wampler II, 2005]
- **parallel implementations** in collaboration with Yusong Wang, Anton Leykin, Yan Zhuang
- since release 2.3.13 PHCpack contains **MixedVol** (ACM TOMS Algorithm 845), developed by Tangan Gao, Tien-Yien Li, Xing Li, and Mengnien Wu
- deflation for **isolated singularities** in collaboration with Anton Leykin and Ailing Zhao
- PHCmaple (Anton Leykin) and PHClab (Yun Guan)

Numerical Homotopy Continuation Methods

are pleasingly parallel

- one job = track one path
- in manager/worker scenario:
  1. manager distributes jobs to idle workers;
  2. workers execute jobs and report to manager.
- dynamic load balancing of jobs is needed

Thanks to Linux, MPI, and cluster computers (Rocketcalc), possible to track many hundreds of thousands of paths.

Other Parallel Homotopy Solvers

H.-J. Su, J.M. McCarthy, M. Sosonkina, and L.T. Watson:
Algorithm 8xx: **POLSYS_GLP**: A parallel general linear product homotopy code for solving polynomial systems of equations. To appear in *ACM Trans. Math. Softw.*

$\rightarrow$ applied to large systems from mechanism design

T. Gunji, S. Kim, K. Fujisawa, and M. Kojima:
PHoMpara – parallel implementation of the Polyhedral Homotopy continuation Method for polynomial systems. *Computing* 77(4):387–411, 2006.

$\rightarrow$ Masakazu Kojima's talk on Wednesday 1:40PM

a problem from electromagnetics

W. Boege, R. Gebauer, and H. Kredel: "Some examples for solving systems of algebraic equations by calculating Groebner bases." *J. Symbolic Computation*, 2:83–98, 1986.

an ancient benchmark problem...

posed by Shigetoshi Katsura to PoSSo in 1994:
a family of $n-1$ **quadrics** and one linear equation.

...but relevant to applications

#solutions is 2^{n-1} (= Bézout bound).

$\rightarrow$ *we have an optimal homotopy!*

The Total Degree Homotopy

A natural choice of a homotopy to solve is for example $h(\mathbf{x},t) =$

$$\gamma \underbrace{\left(\begin{array}{r} x_1^2 - 1 = 0 \\ x_2^2 - 1 = 0 \end{array} \right)}_{\textbf{start system}} (1-t) + \underbrace{\left(\begin{array}{r} x_1^2 + x_2 - 3 = 0 \\ x_1 + 0.125 x_2^2 - 1.5 = 0 \end{array} \right)}_{\textbf{target system}} t = \mathbf{0},$$

where t goes from 0 to 1, and $\gamma \in \mathbb{C}$.

For almost all choices of $\gamma \in \mathbb{C}$, every isolated solution of multiplicity m is reached by exactly m solution paths.

also called "the gamma trick"

(If we take $\gamma = 1$, then at $t \approx 0.92$ singular solutions occur.)

Jumpstarting Homotopies

Problem: huge #paths (e.g.: $> 100{,}000$),

undesirable to store all start solutions in main memory.

Solution: (assume manager/worker protocol)

1. The manager reads start solution from file "**just in time**" whenever a worker needs another path tracking job.
2. For total degree and linear-product start systems, it is **simple to compute** the solutions whenever needed.
3. As soon as worker reports the end of a solution path back to the manager, the solution is **written to file**.

Indexing Start Solutions

The start system $\left\{ \begin{array}{l} x_1^4 - 1 = 0 \\ x_2^5 - 1 = 0 \\ x_3^3 - 1 = 0 \end{array} \right.$ has $4 \times 5 \times 3 = 60$ solutions.

Get 25th solution via decomposition: $24 = 1(5 \times 3) + 3(3) + 0$.

Verify via lexicographic enumeration:

$$000 \rightarrow 001 \rightarrow 002 \rightarrow 010 \rightarrow 011 \rightarrow 012 \rightarrow 020 \rightarrow 021 \rightarrow 022 \rightarrow 030 \rightarrow 031 \rightarrow 032 \rightarrow 040 \rightarrow 041 \rightarrow 042$$

$$100 \rightarrow 101 \rightarrow 102 \rightarrow 110 \rightarrow 111 \rightarrow 112 \rightarrow 120 \rightarrow 121 \rightarrow 122 \rightarrow \boxed{130} \rightarrow 131 \rightarrow 132 \rightarrow 140 \rightarrow 141 \rightarrow 142$$

$$200 \rightarrow 201 \rightarrow 202 \rightarrow 210 \rightarrow 211 \rightarrow 212 \rightarrow 220 \rightarrow 221 \rightarrow 222 \rightarrow 230 \rightarrow 231 \rightarrow 232 \rightarrow 240 \rightarrow 241 \rightarrow 242$$

$$300 \rightarrow 301 \rightarrow 302 \rightarrow 310 \rightarrow 311 \rightarrow 312 \rightarrow 320 \rightarrow 321 \rightarrow 322 \rightarrow 330 \rightarrow 331 \rightarrow 332 \rightarrow 340 \rightarrow 341 \rightarrow 342$$

Using Linear-Product Start Systems Efficiently

- Store start systems in their linear-product product form, e.g.:

$$g(\mathbf{x}) = \left\{ \begin{array}{c} (\cdots) \cdot (\cdots) \cdot (\cdots) \cdot (\cdots) = 0 \\ (\cdots) \cdot (\cdots) \cdot (\cdots) \cdot (\cdots) \cdot (\cdots) = 0 \\ (\cdots) \cdot (\cdots) \cdot (\cdots) = 0 \end{array} \right.$$

- Lexicographic enumeration of start solutions,

  $\longrightarrow$ as many candidates as the total degree.

- Eventually store results of incremental LU factorization.

  $\longrightarrow$ prune in the tree of combinations.

A first run of 1,048,576 paths

dynamic load balancing

on personal cluster computers built by Rocketcalc

1 manager + 13 workers at 2.4Ghz: 32 hours and 44 minutes

#paths per processor: 64,073 `min` 90,504 `max` 80,660 `avg` @UIC

shared memory machine with 16 computational cores

AMD64 opteron at 1.8Ghz: 31 hours and 31 minutes

#paths per processor: 64,659 `min` 74,348 `max` 69,905 `avg` @IMA

verification of output:

1. parsing 1.3Gb file into memory takes 400Mb and 4 minutes;
2. data compression to quadtree of 58Mb takes 7 seconds.

Using the Blackbox solver: phc -b

an input file:

```
8
  z0 + z1 + z2 + z3 + z4 + z5 + z6 + z7;

  z0*z1 + z1*z2 + z2*z3 + z3*z4 + z4*z5 + z5*z6 + z6*z7 + z7*z0;

  z0*z1*z2 + z1*z2*z3 + z2*z3*z4 + z3*z4*z5 + z4*z5*z6 + z5*z6*z7
+ z6*z7*z0 + z7*z0*z1;

  z0*z1*z2*z3 + z1*z2*z3*z4 + z2*z3*z4*z5 + z3*z4*z5*z6
+ z4*z5*z6*z7 + z5*z6*z7*z0 + z6*z7*z0*z1 + z7*z0*z1*z2;

  z0*z1*z2*z3*z4 + z1*z2*z3*z4*z5 + z2*z3*z4*z5*z6 + z3*z4*z5*z6*z7
+ z4*z5*z6*z7*z0 + z5*z6*z7*z0*z1 + z6*z7*z0*z1*z2 + z7*z0*z1*z2*z3;
```

```
  z0*z1*z2*z3*z4*z5 + z1*z2*z3*z4*z5*z6 + z2*z3*z4*z5*z6*z7
+ z3*z4*z5*z6*z7*z0 + z4*z5*z6*z7*z0*z1 + z5*z6*z7*z0*z1*z2
+ z6*z7*z0*z1*z2*z3 + z7*z0*z1*z2*z3*z4;

  z0*z1*z2*z3*z4*z5*z6 + z1*z2*z3*z4*z5*z6*z7 + z2*z3*z4*z5*z6*z7*z0
+ z3*z4*z5*z6*z7*z0*z1 + z4*z5*z6*z7*z0*z1*z2 + z5*z6*z7*z0*z1*z2*z3
+ z6*z7*z0*z1*z2*z3*z4 + z7*z0*z1*z2*z3*z4*z5;

  z0*z1*z2*z3*z4*z5*z6*z7 - 1;

TITLE : cyclic 8-roots problem
```

save as `cyclic8` and then type

```
   phc -b cyclic8 cyclic8.out
```

After 42 minutes and 11 seconds...

on a (slow) Apple laptop

```
solution 2560 :    start residual :  4.012E-15   #iterations : 1   success
t :  1.00000000000000E+00   0.00000000000000E+00
m : 1
the solution for t :
 z0 : -9.10932685974638E-01   4.12555016480266E-01
 z1 : -9.50467926146792E-01  -3.10822652595037E-01
 z2 :  9.62715731458717E-01   2.70515101984913E-01
 z3 :  3.82683432365089E-01   9.23879532511287E-01
 z4 : -4.89459759042523E-01   8.72025885096329E-01
 z5 :  4.52297510482370E-01  -8.91867121275053E-01
 z6 :  9.35847129222866E-01  -3.52406229691418E-01
 z7 : -3.82683432365090E-01  -9.23879532511287E-01
== err :  5.017E-15 = rco :  3.132E-02 = res :  3.967E-15 = complex regular =
```

The Quality of an Approximate Solution

The numbers in

```
== err :  5.017E-15 = rco :  3.132E-02 = res :  3.967E-15 = complex regular =
```

are diagnostics of a Newton iteration:

$$\text{solve} \quad J_f(\mathbf{x})\Delta\mathbf{x} = -f(\mathbf{x}) \quad \text{for } \Delta\mathbf{x}$$

use $\Delta\mathbf{x}$ to update $\mathbf{x}$, i.e.: $\mathbf{x} := \mathbf{x} + \Delta\mathbf{x}$.

$$\texttt{err} = ||\Delta\mathbf{x}|| \quad \texttt{rco} = 1/(\text{cond\# of } J_f(\mathbf{x})) \quad \texttt{res} = ||f(\mathbf{x})||$$

The err $\approx 10^{-15}$ and rco $\approx 10^{-2}$ implies that, with input accurate up to standard double floating precision, we are sure of all except the last two decimal places in the solution vectors.

Frequency Tables of Diagnostics

```
Frequency tables for correction, residual, condition, and distances :
FreqCorr :  737 42 31 21 1 512 64 0 0 0 0 0 0 57 0 1095 : 2560
FreqResi :  331 78 321 100 2 0 0 0 0 0 0 324 243 306 0 855 : 2560
FreqCond :  128 753 209 62 0 0 0 141 339 78 18 0 0 0 0 832 : 2560
FreqDist :  1984 0 0 0 0 312 264 0 0 0 0 0 0 0 0 0 : 2560
Small correction terms and residuals counted to the right.
Well conditioned and distinct roots counted to the left.
```

Counting solutions with small $||\Delta \mathbf{x}||$ and small condition number:

```
57 + 1095 = 1152
128 + 753 + 209 + 62 = 1152
```

The well conditioned roots are well separated from the points on the 1-dimensional solution curve; #isolated solutions is 1152.

Mixed Volumes and Polyhedral Homotopies

input: $f(\mathbf{x}) = \mathbf{0}$ a system of n equations in n unknowns

- **Stage 1**: compute the **mixed volume V**

  input: $\mathcal{Q} = (Q_1, Q_2, \ldots, Q_n)$ Newton polytopes of f
  output: a **regular mixed-cell configuration Δ_ω**: $V = \sum_{C \in \Delta_\omega} \mathrm{vol}(C)$

- **Stage 2**: solve a random coefficient start system $g(\mathbf{x}) = \mathbf{0}$

  input: a **regular mixed-cell configuration Δ_ω**
  output: $g(\mathbf{x}) = \mathbf{0}$ a random coefficient start system: $\#g^{-1}(\mathbf{0}) = V$

- **Stage 3**: use $h(\mathbf{x}, t) = g(\mathbf{x})(1-t) + f(\mathbf{x})t = \mathbf{0}$ to solve f

output: approximations to all isolated solutions of f in $(\mathbb{C}^*)^n \leq V$

Computing Mixed Volumes with phc -m

```
getafix:~/PHCv2/Demo jan$ phc -m
Welcome to PHC (Polynomial Homotopy Continuation) V2.3.15 28 Sep 2006
Mixed-Volume Computation by various lifting strategies and MixedVol.
...
MENU with available Lifting Strategies (0 is default) :
  0. Static lifting     : lift points and prune lower hull.
  1. Implicit lifting   : based on recursive formula.
  2. Dynamic lifting    : incrementally add the points.
  3. Symmetric lifting  : points in same orbit get same lifting.
  4. MixedVol Algorithm : a faster mixed volume computation.
Type 0, 1, 2, 3, or 4 to select, eventually preceded by i for info :
```

Why bother?

timings for `phc -b cyclic8 cyclic8.out`:

compute mixed volume	:	27 sec 850 ms
solve start system	:	13 min 41 sec 550 ms
solve target system	:	27 min 25 sec 680 ms
total cpu time	:	42 min 11 sec 160 ms

old version of `phc`, without MixedVol: 56 sec 560 ms

to compute the mixed volume

Why bother?

timings for `phc -b cyclic8 cyclic8.out`:

compute mixed volume	:	27 sec 850 ms
solve start system	:	13 min 41 sec 550 ms
solve target system	:	27 min 25 sec 680 ms
total cpu time	:	42 min 11 sec 160 ms

old version of `phc`, without MixedVol: 56 sec 560 ms

to compute 2560 as the mixed volume

However, the "old" `phc` could not compute the mixed volume of cyclic 11-roots: 184,756...

...which takes 24 min 22 sec 690 ms on same Apple laptop.

Numerical Irreducible Decomposition

input: $f(\mathbf{x}) = \mathbf{0}$ a polynomial system with $\mathbf{x} \in \mathbb{C}^n$

- **Stage 1**: represent the k-dimensional solutions Z_k, $k = 0, 1, \ldots$

  output: sequence $[W_0, W_1, \ldots, W_{n-1}]$ of ***witness sets***
  $W_k = (E_k, E_k^{-1}(\mathbf{0}) \setminus J_k)$, $\deg Z_k = \#(E_k^{-1}(\mathbf{0}) \setminus J_k)$
  $E_k = f + k$ random hyperplanes, $J_k =$ “junk”

- **Stage 2**: decompose Z_k, $k = 0, 1, \ldots$ into irreducible factors

  output: $W_k = \{W_{k1}, W_{k2}, \ldots, W_{kn_k}\}$, $k = 1, 2, \ldots, n-1$
  n_k irreducible components of dimension k

output: a numerical irreducible decomposition of $f^{-1}(\mathbf{0})$
is a sequence of partitioned witness sets

Computing Witness Sets for $f^{-1}(\mathbf{0})$

Witness set $W_k = (E_k, E_k^{-1}(\mathbf{0}) \setminus J_k)$ for $Z_k \subset f^{-1}(\mathbf{0})$, $k = \dim Z_k$,
consists of $E_k = f + k$ random hyperplanes
and its solutions, $\#(E_k^{-1}(\mathbf{0}) \setminus J_k) = \deg Z_k$.

- **top down**: use a cascade of homotopies
  - $+$ benefits from existing blackbox solver
  - $-$ requires top dimension on input
- **bottom up**: with an equation-by-equation solver
  - $+$ requires no guess for top dimension
  - $-$ performance depends on order of equations

Example of a Homotopy in the Cascade

To compute numerical representations of the twisted cubic and the four isolated points, as given by the solution set of one polynomial system, we use the following homotopy:

$$H(\mathbf{x}, z_1, t) = \left[\begin{array}{c} \left[\begin{array}{c} (x_1^2 - x_2)(x_1 - 0.5) \\ (x_1^3 - x_3)(x_2 - 0.5) \\ (x_1 x_2 - x_3)(x_3 - 0.5) \end{array} \right] + t \left[\begin{array}{c} \gamma_1 \\ \gamma_2 \\ \gamma_3 \end{array} \right] z_1 \\ t\,(c_0 + c_1 x_1 + c_2 x_2 + c_3 x_3) + z_1 \end{array} \right] = \mathbf{0}$$

At $t = 1$: $H(\mathbf{x}, z_1, t) = \mathcal{E}(f)(\mathbf{x}, z_1) = \mathbf{0}$.

At $t = 0$: $H(\mathbf{x}, z_1, t) = f(\mathbf{x}) = \mathbf{0}$.

As t goes from 1 to 0, the hyperplane is removed from the system, and z_1 is forced to zero.

A Cascade of Homotopies

Denote $\mathcal{E}_i$ as an embedding of $f(\mathbf{x}) = \mathbf{0}$ with i random hyperplanes and i slack variables $\mathbf{z} = (z_1, z_2, \ldots, z_i)$.

Theorem (Sommese - Verschelde): *J. Complexity* 16(3):572–602, 2000

1. Solutions with $(z_1, z_2, \ldots, z_i) = \mathbf{0}$ contain $\deg W$ generic points on every i-dimensional component W of $f(\mathbf{x}) = \mathbf{0}$.
2. Solutions with $(z_1, z_2, \ldots, z_i) \neq \mathbf{0}$ are regular; and

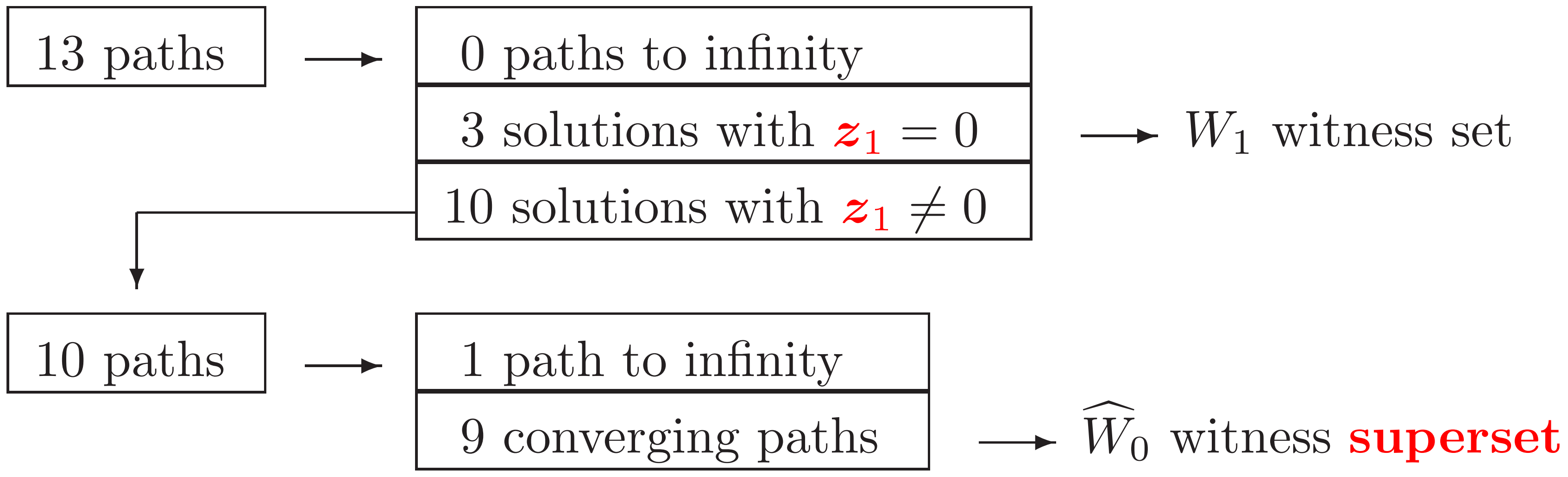
solution paths defined by

$$H_i(\mathbf{x}, \mathbf{z}, t) = t\mathcal{E}_i(\mathbf{x}, \mathbf{z}) + (1-t)\begin{pmatrix} \mathcal{E}_{i-1}(\mathbf{x}, \mathbf{z}) \\ z_i \end{pmatrix} = \mathbf{0}$$

starting at $t = 1$ with all solutions with $z_i \neq 0$ reach at $t = 0$ all isolated solutions of $\mathcal{E}_{i-1}(\mathbf{x}, \mathbf{z}) = \mathbf{0}$.

#paths in twisted cubic + 4 isolated points example

The flow chart below summarizes the number of solution paths traced in the cascade of homotopies.


13 paths
0 paths to infinity
3 solutions with $z_1 = 0$
W_1 witness set
10 solutions with $z_1 \neq 0$
10 paths
1 path to infinity
9 converging paths
$\widehat{W}_0$ witness superset

The set $\widehat{W}_0$ contains, in addition to the four isolated roots, also points on the twisted cubic. The points in $\widehat{W}_0$ which lie on the twisted cubic are considered **junk** and must be filtered out.

Computing Witness Sets in phc

```
3
 (x1^2 - x2)*(x1 - 0.5);                     save as
 (x1^3 - x3)*(x2 - 0.5);                    "twistp4i"
 (x1*x2 - x3)*(x3 - 0.5);
```

top down:

- `phc -c` add 1 random hyperplane, save as `"twistp4i_e1"`.
- `phc -c` run cascade, extract from output witness set `"twistp4i_ws1"`, and candidate isolated points `"twistp4i_sw0"`.
- `phc -f` filter `"twistp4i_sw0"` with respect to `"twistp4i_ws1"`.

bottom up:

- `phc -a` runs equation-by-equation solver.
- `phc -w` witness set intersection using diagonal homotopies.

Factorization Methods in phc

```
3  8
  x11*x22 - x21*x12;                    save as
  x12*x23 - x22*x13;                   "minors24"
  x13*x24 - x23*x14;
```

- `phc -c`   add 5 random hyperplanes, save as `"minors24e5"`.
- `phc -b`   solve `"minors24e5"`.
- `phc -f`   factor using monodromy breakup certified by linear traces.

  or   factor using combinatorial enumeration of linear traces.

Deflation for Isolated Singular Solutions

input: $f(\mathbf{x}) = \mathbf{0}$ a polynomial system;
$\mathbf{x}^*$ an approximate solution: $f(\mathbf{x}^*) \approx \mathbf{0}$

- **Stage 1**: recondition the problem

  output: $G(\mathbf{x}, \boldsymbol{\lambda}) = \mathbf{0}$ an extension to f;
  $(\mathbf{x}^*, \boldsymbol{\lambda}^*)$ is a **regular** solution: $G(\mathbf{x}^*, \boldsymbol{\lambda}^*) = \mathbf{0} \Rightarrow f(\mathbf{x}^*) = \mathbf{0}$.

- **Stage 2**: compute the multiplicity

  input: $g(\mathbf{x}) = \mathbf{0}$ a "good" system for $\mathbf{x}^*$
  Newton converges quadratically starting from $\mathbf{x}^*$.

  Apply algorithms of [Dayton & Zeng, 2005] (in `phc -v`)
  or [Bates, Peterson & Sommese, 2006].

output: a quadratically convergent method to refine $\mathbf{x}^*$
and the multiplicity structure of $\mathbf{x}^*$

Deflation Operator Dfl reduces to Corank One

Consider $f(\mathbf{x}) = \mathbf{0}$, N equations in n unknowns, $N \geq n$.

Suppose $\mathrm{Rank}(A(\mathbf{z}_0)) = R < n$ for $\mathbf{z}_0$ an isolated zero of $f(\mathbf{x}) = 0$.

Choose $\mathbf{h} \in \mathbb{C}^{R+1}$ and $B \in \mathbb{C}^{n \times (R+1)}$ at random.

Introduce $R + 1$ new multiplier variables $\boldsymbol{\lambda} = (\lambda_1, \lambda_2, \ldots, \lambda_{R+1})$.

$$\mathbf{Dfl}(f)(\mathbf{x}, \boldsymbol{\lambda}) := \left\{ \begin{array}{rcl} f(\mathbf{x}) & = & \mathbf{0} \\ A(\mathbf{x})B\boldsymbol{\lambda} & = & \mathbf{0} \\ \mathbf{h}\boldsymbol{\lambda} & = & 1 \end{array} \right. \qquad \begin{array}{c} \mathrm{Rank}(A(\mathbf{x})) = R \\ \Downarrow \\ \mathrm{corank}(A(\mathbf{x})B) = 1 \end{array}$$

The operator **Dfl** is used recursively if necessary, note:

(1) # times bounded by multiplicity

(2) symbolic implementation easy, but leads to expression swell

(3) exploiting the structure for evaluation is efficient

Newton's Method with Deflation

Input: $f(\mathbf{x}) = \mathbf{0}$ polynomial system;
$\mathbf{x}_0$ initial approximation for $\mathbf{x}^*$;
ϵ tolerance for numerical rank.

Newton's Method with Deflation

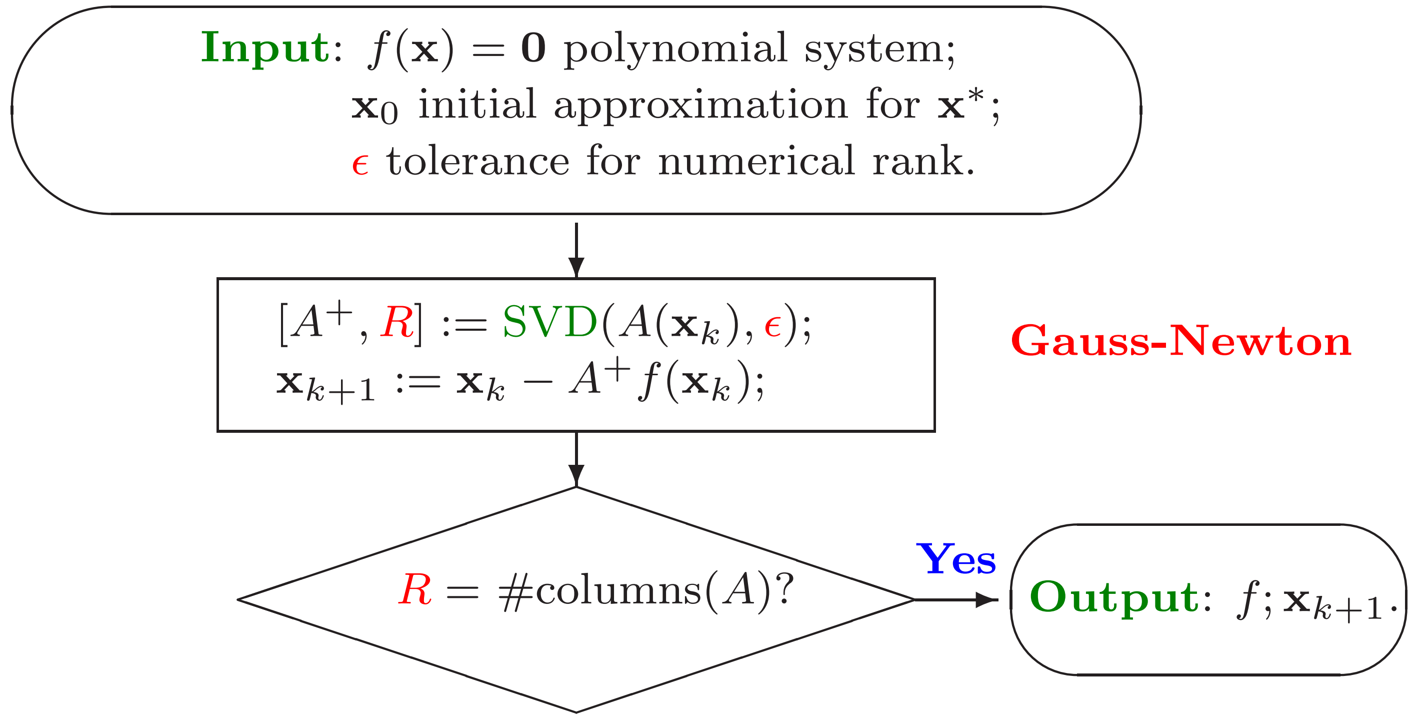
Input: $f(\mathbf{x}) = \mathbf{0}$ polynomial system;
$\mathbf{x}_0$ initial approximation for $\mathbf{x}^*$;
ϵ tolerance for numerical rank.

$$\downarrow$$

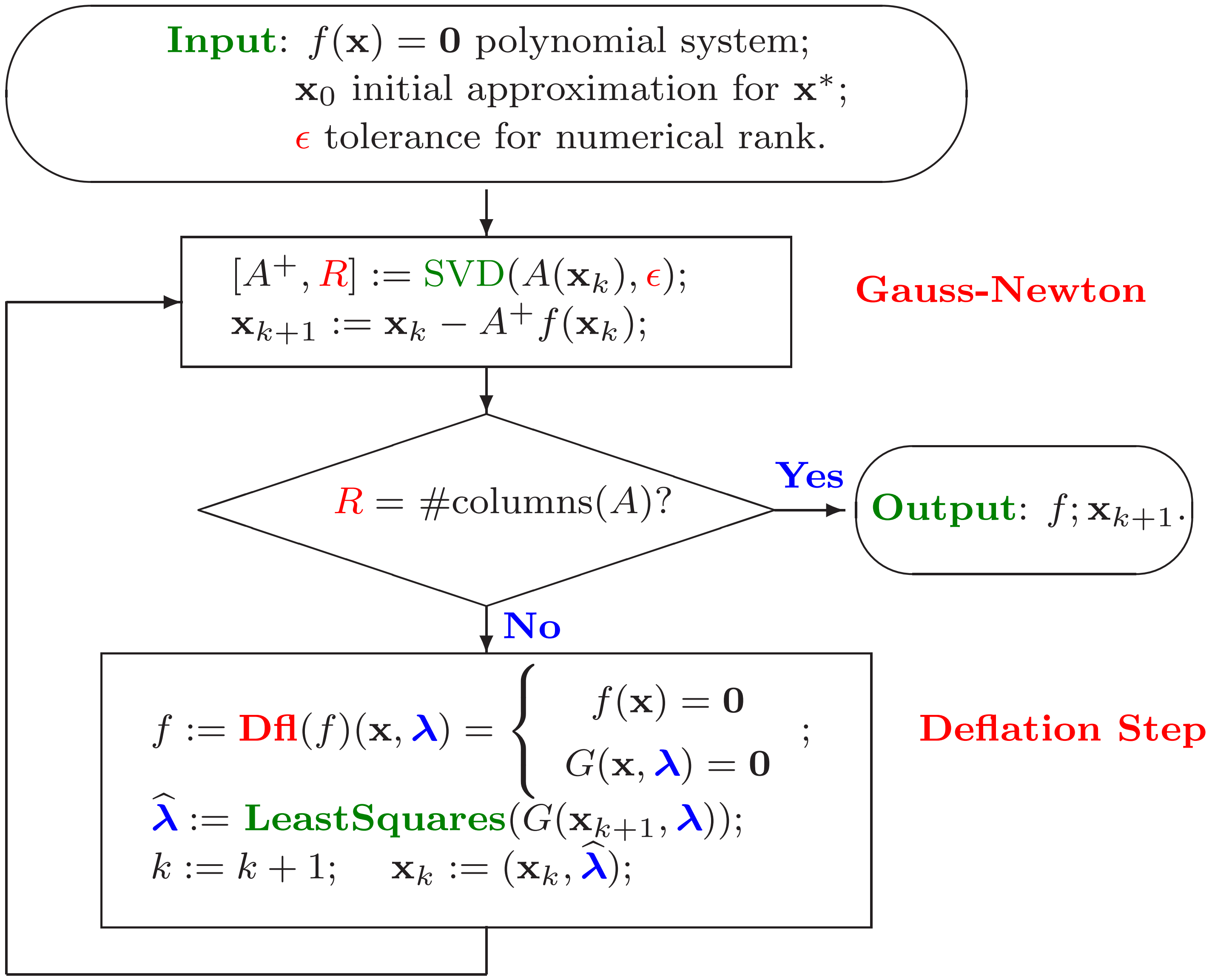
$$\boxed{\begin{array}{l} [A^+, R] := \mathrm{SVD}(A(\mathbf{x}_k), \epsilon); \\ \mathbf{x}_{k+1} := \mathbf{x}_k - A^+ f(\mathbf{x}_k); \end{array}}$$

Gauss-Newton

Newton's Method with Deflation


Input: $f(\mathbf{x}) = \mathbf{0}$ polynomial system;
$\mathbf{x}_0$ initial approximation for $\mathbf{x}^*$;
ϵ tolerance for numerical rank.
$[A^+, R] := \mathrm{SVD}(A(\mathbf{x}_k), \epsilon)$;
$\mathbf{x}_{k+1} := \mathbf{x}_k - A^+ f(\mathbf{x}_k)$;
Gauss-Newton
$R = \#\mathrm{columns}(A)$?
Yes
Output: $f; \mathbf{x}_{k+1}$.

Newton's Method with Deflation


Input: $f(\mathbf{x}) = \mathbf{0}$ polynomial system;
$\mathbf{x}_0$ initial approximation for $\mathbf{x}^*$;
ϵ tolerance for numerical rank.
$[A^+, R] := \mathrm{SVD}(A(\mathbf{x}_k), \epsilon)$;
$\mathbf{x}_{k+1} := \mathbf{x}_k - A^+ f(\mathbf{x}_k)$;
Gauss-Newton
$R = \#\mathrm{columns}(A)$?
Yes
Output: $f; \mathbf{x}_{k+1}$.
No
$f := \mathbf{Dfl}(f)(\mathbf{x}, \boldsymbol{\lambda}) = \left\{ \begin{array}{c} f(\mathbf{x}) = \mathbf{0} \\ G(\mathbf{x}, \boldsymbol{\lambda}) = \mathbf{0} \end{array} \right.$;
$\widehat{\boldsymbol{\lambda}} := \mathbf{LeastSquares}(G(\mathbf{x}_{k+1}, \boldsymbol{\lambda}))$;
$k := k + 1; \quad \mathbf{x}_k := (\mathbf{x}_k, \widehat{\boldsymbol{\lambda}})$;
Deflation Step

Avoiding Expression Swell

Evaluation of $A(\mathbf{x})B$: for efficiency we must first replace $\mathbf{x}$ by values ***before*** the matrix multiplication.

Triangular block structure of Jacobian matrix: for example:

$$A_2(\mathbf{x}, \boldsymbol{\lambda}_1, \boldsymbol{\lambda}_2) = \begin{bmatrix} A & \mathbf{0} & \mathbf{0} \\ \left(\frac{\partial A}{\partial \mathbf{x}}\right) B_1 \boldsymbol{\lambda}_1 & A B_1 & \mathbf{0} \\ \mathbf{0} & \mathbf{h}_1 & \mathbf{0} \\ \left(\frac{\partial A_1}{\partial \mathbf{x}}\right) B_2 \boldsymbol{\lambda}_2 & \left(\frac{\partial A_1}{\partial \boldsymbol{\lambda}_1}\right) B_2 \boldsymbol{\lambda}_2 & A_1 B_2 \\ \mathbf{0} & \mathbf{0} & \mathbf{h}_2 \end{bmatrix}.$$

Multipliers occur linearly: compute derivatives only with respect to $\mathbf{x}$, not with respect to $\boldsymbol{\lambda}$.

Structure of the Deflated Systems

At stage k in the deflation:

$$f_k(\mathbf{x}, \boldsymbol{\lambda}_1, \ldots, \boldsymbol{\lambda}_{k-1}, \boldsymbol{\lambda}_k) = \left\{ \begin{array}{rcl} f_{k-1}(\mathbf{x}, \boldsymbol{\lambda}_1, \ldots, \boldsymbol{\lambda}_{k-1}) & = & 0 \\ A_{k-1}(\mathbf{x}, \boldsymbol{\lambda}_1, \ldots, \boldsymbol{\lambda}_{k-1}) B_k \boldsymbol{\lambda}_k & = & 0 \\ \mathbf{h}_k \boldsymbol{\lambda}_k & = & 1, \end{array} \right.$$

$f_0 = f$, $A_0 = A$, $R_k = \mathrm{rank}(A_{k-1}(\mathbf{z}_0))$, $R_k + 1$ multipliers in $\boldsymbol{\lambda}_k$.

random vector $\mathbf{h}_k \in \mathbb{C}^{R_k+1}$ and n_{k-1}-by-$(R_k + 1)$ matrix B_k

$$\begin{array}{rcl} \#\text{rows in } A_k & : & N_k = 2N_{k-1} + 1, \quad N_0 = N, \\ \#\text{columns in } A_k & : & n_k = n_{k-1} + R_k + 1, \quad n_0 = n. \end{array}$$

Multiplication of the polynomial matrix A_{k-1} with the random matrix B_k and vector of $R_k + 1$ multiplier variables $\boldsymbol{\lambda}_k$ is expensive!

Structure of the Jacobian Matrices

At stage k in the deflation:

$$f_k(\mathbf{x}, \boldsymbol{\lambda}_1, \ldots, \boldsymbol{\lambda}_{k-1}, \boldsymbol{\lambda}_k) = \left\{ \begin{array}{rcl} f_{k-1}(\mathbf{x}, \boldsymbol{\lambda}_1, \ldots, \boldsymbol{\lambda}_{k-1}) & = & 0 \\ A_{k-1}(\mathbf{x}, \boldsymbol{\lambda}_1, \ldots, \boldsymbol{\lambda}_{k-1}) B_k \boldsymbol{\lambda}_k & = & 0 \\ \mathbf{h}_k \boldsymbol{\lambda}_k & = & 1. \end{array} \right.$$

Jacobian matrix at the kth deflation:

$$A_k(\mathbf{x}, \boldsymbol{\lambda}_1, \ldots, \boldsymbol{\lambda}_{k-1}, \boldsymbol{\lambda}_k) = \left[\begin{array}{cc} A_{k-1} & \mathbf{0} \\ \left[\frac{\partial A_{k-1}}{\partial \mathbf{x}} \frac{\partial A_{k-1}}{\partial \boldsymbol{\lambda}_1} \cdots \frac{\partial A_{k-1}}{\partial \boldsymbol{\lambda}_{k-1}} \right] B_k \boldsymbol{\lambda}_k & A_{k-1} B_k \\ \mathbf{0} & \mathbf{h}_k \end{array} \right].$$

The multiplier variables $\boldsymbol{\lambda}_i$, $i = 1, 2, \ldots, k$, occur linearly,

we compute derivatives only with respect to the original variables $\mathbf{x}$.

Derivatives of Jacobian matrices

Define $\frac{\partial A}{\partial \mathbf{x}} = \left[\frac{\partial A}{\partial x_1} \ \frac{\partial A}{\partial x_2} \ \cdots \ \frac{\partial A}{\partial x_n} \right]$, $\mathbf{x} = (x_1, x_2, \ldots, x_n)$,

where $\frac{\partial A}{\partial x_k} = \left[\frac{\partial a_{ij}}{\partial x_k} \right]$ for $A = [a_{ij}(\mathbf{x})]_{j,k \in \{1,2,\ldots,n\}}^{i \in \{1,2,\ldots,N\}}$.

Natural tree structure:
(to compute $\frac{\partial^2 A}{\partial \mathbf{x}^2}$)

$$\begin{tikzcd}
& & & A(x_1, x_2) \arrow[dll, dash] \arrow[drr, dash] & & & \\
& \frac{\partial A}{\partial x_1} \arrow[dl, dash] \arrow[dr, dash] & & & & \frac{\partial A}{\partial x_2} \arrow[dl, dash] \arrow[dr, dash] & \\
\frac{\partial^2 A}{\partial x_1^2} & & \frac{\partial^2 A}{\partial x_2 \partial x_1} & = & \frac{\partial^2 A}{\partial x_1 \partial x_2} & & \frac{\partial^2 A}{\partial x_2^2}
\end{tikzcd}$$

Complexity of $\frac{\partial^k A}{\partial \mathbf{x}^k} \neq O(n^k)$,

but O(#monomials of degree k in n variables).

$$\boxed{\begin{array}{l} \text{e.g.: } k = 10,\ n = 3: \\ \qquad 3^{10} = 59049 \gg 66 \end{array}}$$

Column Format of Jacobian Matrices

Jacobian matrix at the kth deflation:

$$A_k(\mathbf{x}, \boldsymbol{\lambda}_1, \ldots, \boldsymbol{\lambda}_{k-1}, \boldsymbol{\lambda}_k) = \left[\begin{array}{cc} A_{k-1} & \mathbf{0} \\ \left[\frac{\partial A_{k-1}}{\partial \mathbf{x}} \; \frac{\partial A_{k-1}}{\partial \boldsymbol{\lambda}_1} \cdots \frac{\partial A_{k-1}}{\partial \boldsymbol{\lambda}_{k-1}} \right] B_k \boldsymbol{\lambda}_k & A_{k-1} B_k \\ \mathbf{0} & \mathbf{h}_k \end{array} \right].$$

Tree with k children:

$$\begin{tikzcd}
 & & A_k(\mathbf{x}, \boldsymbol{\lambda}_1, \ldots, \boldsymbol{\lambda}_{k-1}, \boldsymbol{\lambda}_k) \arrow[dll] \arrow[dl] \arrow[d] \arrow[drr] & & \\
A_{k-1} & \frac{\partial A_{k-1}}{\partial \mathbf{x}} & \frac{\partial A_{k-1}}{\partial \boldsymbol{\lambda}_1} & \cdots & \frac{\partial A_{k-1}}{\partial \boldsymbol{\lambda}_{k-1}}
\end{tikzcd}$$

Unwinding the Multipliers

$$A_2(\mathbf{x}, \boldsymbol{\lambda}_1, \boldsymbol{\lambda}_2) = \begin{bmatrix} A_1 & \mathbf{0} \\ \left[\frac{\partial A_1}{\partial \mathbf{x}} \;\; \frac{\partial A_1}{\partial \boldsymbol{\lambda}_1}\right] B_2 \boldsymbol{\lambda}_2 & A_1 B_2 \\ \mathbf{0} & \mathbf{h}_2 \end{bmatrix}$$

$$\begin{tikzcd} & A_2 \arrow[dl] \arrow[d] \arrow[dr] & \\ A_1 & \frac{\partial A_1}{\partial \mathbf{x}} & \frac{\partial A_1}{\partial \boldsymbol{\lambda}_1} \end{tikzcd}$$

$$\begin{bmatrix} A & \mathbf{0} \\ \left[\frac{\partial A}{\partial \mathbf{x}}\right] B_1 \boldsymbol{\lambda}_1 & A B_1 \\ \mathbf{0} & \mathbf{h}_1 \end{bmatrix} \qquad \begin{bmatrix} \frac{\partial A}{\partial \mathbf{x}} & \mathbf{0} \\ \left[\frac{\partial^2 A}{\partial \mathbf{x}^2}\right] B_1 \boldsymbol{\lambda}_1 & \left[\frac{\partial A}{\partial \mathbf{x}}\right] B_1 \\ \mathbf{0} & \mathbf{0} \end{bmatrix} \qquad \begin{bmatrix} \mathbf{0} & \mathbf{0} \\ \left[\frac{\partial A}{\partial \mathbf{x}}\right] * B_1 & \mathbf{0} \\ \mathbf{0} & \mathbf{0} \end{bmatrix}$$

Unwinding the Multipliers

$$A_2(\mathbf{x},\boldsymbol{\lambda}_1,\boldsymbol{\lambda}_2) = \begin{bmatrix} A_1 & \mathbf{0} \\ \left[\frac{\partial A_1}{\partial \mathbf{x}} \;\; \frac{\partial A_1}{\partial \boldsymbol{\lambda}_1}\right] B_2 \boldsymbol{\lambda}_2 & A_1 B_2 \\ \mathbf{0} & \mathbf{h}_2 \end{bmatrix}$$

Directed Acyclic Graph:

$$\begin{tikzcd}
 & A_2 \arrow[dl] \arrow[d] \arrow[dr] & \\
A_1 \arrow[d] \arrow[dr] & \frac{\partial A_1}{\partial \mathbf{x}} \arrow[d] \arrow[dr] & \frac{\partial A_1}{\partial \boldsymbol{\lambda}_1} \arrow[dl] \\
A & \frac{\partial A}{\partial \mathbf{x}} & \frac{\partial^2 A}{\partial \mathbf{x}^2}
\end{tikzcd}$$

$$\begin{bmatrix} A & \mathbf{0} \\ \left[\frac{\partial A}{\partial \mathbf{x}}\right] B_1 \boldsymbol{\lambda}_1 & A B_1 \\ \mathbf{0} & \mathbf{h}_1 \end{bmatrix} \quad \begin{bmatrix} \frac{\partial A}{\partial \mathbf{x}} & \mathbf{0} \\ \left[\frac{\partial^2 A}{\partial \mathbf{x}^2}\right] B_1 \boldsymbol{\lambda}_1 & \left[\frac{\partial A}{\partial \mathbf{x}}\right] B_1 \\ \mathbf{0} & \mathbf{0} \end{bmatrix} \quad \begin{bmatrix} \mathbf{0} & \mathbf{0} \\ \left[\frac{\partial A}{\partial \mathbf{x}}\right] * B_1 & \mathbf{0} \\ \mathbf{0} & \mathbf{0} \end{bmatrix}$$

The Operator $*$

For a matrix B: $\left[\frac{\partial A}{\partial \mathbf{x}}\right] B = \left[\frac{\partial A}{\partial x_1} B \;\; \frac{\partial A}{\partial x_2} B \;\; \cdots \;\; \frac{\partial A}{\partial x_n} B\right]$.

However, $\frac{\partial}{\partial \boldsymbol{\lambda}} \left(\left[\frac{\partial A}{\partial \mathbf{x}}\right] B \boldsymbol{\lambda} \right) = \left[\underbrace{\frac{\partial A}{\partial \mathbf{x}} \mathbf{b}_1}_{\partial \lambda_1} \;\; \underbrace{\frac{\partial A}{\partial \mathbf{x}} \mathbf{b}_2}_{\partial \lambda_2} \;\; \cdots \;\; \underbrace{\frac{\partial A}{\partial \mathbf{x}} \mathbf{b}_m}_{\partial \lambda_m} \right]$,

$$\boldsymbol{\lambda} = (\lambda_1, \lambda_2, \ldots, \lambda_m), \; B = [\mathbf{b}_1 \;\; \mathbf{b}_2 \;\; \cdots \mathbf{b}_m].$$

Unlike scalar differentiation: $\frac{\partial}{\partial \boldsymbol{\lambda}} \left(\left[\frac{\partial A}{\partial \mathbf{x}}\right] B \boldsymbol{\lambda} \right) \neq \left[\frac{\partial A}{\partial \mathbf{x}}\right] B$.

With the operator $*$ we permute $\left[\frac{\partial A}{\partial \mathbf{x}}\right] B$ into $\frac{\partial}{\partial \boldsymbol{\lambda}} \left(\left[\frac{\partial A}{\partial \mathbf{x}}\right] B \boldsymbol{\lambda} \right)$.

So we have: $\frac{\partial}{\partial \boldsymbol{\lambda}} \left(\left[\frac{\partial A}{\partial \mathbf{x}}\right] B \boldsymbol{\lambda} \right) = \left[\frac{\partial A}{\partial \mathbf{x}}\right] * B$.

A Directed Acyclic Graph of Derivative Operators

$$\begin{tikzcd}
 & & A_3(\mathbf{x}, \boldsymbol{\lambda}_1, \boldsymbol{\lambda}_2, \boldsymbol{\lambda}_3) \arrow[dll] \arrow[dl] \arrow[d] \arrow[dr] & & \\
A_2(\mathbf{x}, \boldsymbol{\lambda}_1, \boldsymbol{\lambda}_2) \arrow[d] \arrow[dr] \arrow[drr] & \frac{\partial A_2}{\partial \mathbf{x}} \arrow[drr] \arrow[drrr] & \frac{\partial A_2}{\partial \boldsymbol{\lambda}_1} \arrow[d] \arrow[drr] & \frac{\partial A_2}{\partial \boldsymbol{\lambda}_2} \arrow[dll] \arrow[dl] & \\
A_1(\mathbf{x}, \boldsymbol{\lambda}_1) \arrow[d] \arrow[dr] & \frac{\partial A_1}{\partial \mathbf{x}} \arrow[d] \arrow[dr] & \frac{\partial A_1}{\partial \boldsymbol{\lambda}_1} \arrow[dl] & \frac{\partial^2 A_1}{\partial \mathbf{x}^2} \arrow[dl] \arrow[d] & \frac{\partial^2 A_1}{\partial \mathbf{x} \partial \boldsymbol{\lambda}_1} \arrow[dll] \\
A(\mathbf{x}) & \frac{\partial A}{\partial \mathbf{x}} & \frac{\partial^2 A}{\partial \mathbf{x}^2} & \frac{\partial^3 A}{\partial \mathbf{x}^3} &
\end{tikzcd}$$

Validation module in phc -v

```
verschel@citrus % /tmp/phc -v
Welcome to PHC (Polynomial Homotopy Continuation) V2.3.15 28 Sep 2006
Validation, refinement and purification of computed solution lists.

MENU with Validation Methods :
  0. Scanning (huge) solution files and creating condition tables;
  1. Basic Validation : refining and weeding out the solution set;
  2. Evaluation of the residuals using multi-precision arithmetic;
  3. Newton's method using multi-precision arithmetic;
  4. Winding-Number Computation by homotopy continuation;
  5. Polyhedral Validation : frequency table of path directions;
  6. Newton's method with deflation for isolated singularities;
  7. Multiplicity structure of isolated singular solutions.
Type 0, 1, 2, 3, 4, 5, 6, or 7 to select, or i for info :
```

Numerical Results (double float)

System	n	m	D	corank($A(\mathbf{x})$)	Inverse Condition#	#Digits
baker1	2	2	1	$1 \rightarrow 0$	1.7e-08 $\rightarrow$ 3.8e-01	$9 \rightarrow 24$
cbms1	3	11	1	$3 \rightarrow 0$	4.2e-05 $\rightarrow$ 5.0e-01	$5 \rightarrow 20$
cbms2	3	8	1	$3 \rightarrow 0$	1.2e-08 $\rightarrow$ 5.0e-01	$8 \rightarrow 18$
mth191	3	4	1	$2 \rightarrow 0$	1.3e-08 $\rightarrow$ 3.5e-02	$7 \rightarrow 13$
decker1	2	3	2	$1 \rightarrow 1 \rightarrow 0$	3.4e-10 $\rightarrow$ 2.6e-02	$6 \rightarrow 11$
decker2	2	4	3	$1 \rightarrow 1 \rightarrow 1 \rightarrow 0$	4.5e-13 $\rightarrow$ 6.9e-03	$5 \rightarrow 16$
decker3	2	2	1	$1 \rightarrow 0$	4.6e-08 $\rightarrow$ 2.5e-02	$8 \rightarrow 17$
ojika1	2	3	2	$1 \rightarrow 1 \rightarrow 0$	9.3e-12 $\rightarrow$ 4.3e-02	$5 \rightarrow 12$
ojika2	3	2	1	$1 \rightarrow 0$	3.3e-08 $\rightarrow$ 7.4e-02	$6 \rightarrow 14$
ojika3	3	2	1	$1 \rightarrow 0$	1.7e-08 $\rightarrow$ 9.2e-03	$7 \rightarrow 15$
		4	1	$2 \rightarrow 0$	6.5e-08 $\rightarrow$ 8.0e-02	$6 \rightarrow 13$
ojika4	3	3	2	$1 \rightarrow 1 \rightarrow 0$	1.9e-13 $\rightarrow$ 2.4e-04	$6 \rightarrow 11$
cyclic9	9	4	1	$2 \rightarrow 0$	5.6e-10 $\rightarrow$ 1.8e-03	$5 \rightarrow 15$