

Appendix A

POLSYS_GLP User's Guide

A.1 Compilation and Usage

System requirements:

Compiler: Fortran 90, e.g. Portland Group Fortran 90 Compiler
Software Packages: MPI-2
Systems: Linux, Unix that support MPI-2.

File list:

Fortran code: main_template.f90, polsys_glp.f90, lapack_glp.f, test_install.f90
Others: REAMDE, INPUT.DAT, OUTPUT.DAT

Sample makefile:

```
polysys_glp:
    pgf90 -c lapack_glp.f
    pgf90 -c polsys_glp.f90
    pgf90 -c main_template.f90
    mpif90 lapack_glp.o polsys_glp.o main_template.o -o POLSYS\_GLP
    rm *.o *.mod
```

Note the file extension may need to be changed for some compilers.

Usage:

```
mpirun -np 4 POLSYS\_GLP
```

requests four processors to launch POLSYS_GLP program. The file INPUT.DAT should be located in the same directory as the program. When the program exits, an OUTPUT.DAT file with solution data is generated.

A.2 README File

POLSYS_GLP

POLSYS_GLP is Fortran 95 parallel code for solving N complex coefficient polynomial systems of equations in N unknowns by a probability-one, globally convergent homotopy method. The package is fully portable across various distributed and parallel computing platforms since it uses the Message Passing Interface (MPI) communication library. POLSYS_GLP consists of two modules: GLOBAL_GLP and POLSYS2. The module GLOBAL_GLP contains the derived data types that define the polynomial system, the system covering, and the start system of the homotopy; the module POLSYS2 contains the actual solver POLSYS_GLP and its internal routines, CHECK_GLP that verifies the input set covering structure, and the routines responsible for root counting, BEZOUT_GLP and SINGSYS_GLP. POLSYS_GLP uses the HOMPACT90 modules HOMOTOPY, HOMPACT90_GLOBAL, and REAL_PRECISION, the HOMPACT90 path tracking routine STEPNX, and numerous BLAS and LAPACK routines.

The physical organization into files is: the file polsys_glp.f90 contains

(in order) REAL_PRECISION, GLOBAL_GLP, POLSYS2, HOMPACT90_GLOBAL, HOMOTOPY, and STEPNX; the file lapack_glp.f contains all the necessary BLAS and LAPACK routines. A sample calling program MAIN_TEMPLATE and a template for a hand-crafted function/Jacobian matrix evaluation program TARGET_SYSTEM_USER are contained in the file main_template.f90. All processors execute MAIN_TEMPLATE and read the data file INPUT.DAT. MAIN_TEMPLATE writes the solutions to the file OUTPUT.DAT only by the master processor. The included files INPUT.DAT and OUTPUT.DAT contain sample input and output data for several test polynomial systems. Similar to POLSYS_PLP (Algorithm 801), POLSYS_GLP can batch process multiple systems and tasks from one input file. The file test_install.f90 contains a main program TEST_INSTALL to verify the installation. It reads INPUT.DAT, solves a problem defined there, compares the computed results to known answers, and prints a message indicating whether the installation was apparently successful.

To test the package, compile polsys_glp.f90 (as free form Fortran 95 files) and compile lapack_glp.f (as fixed form Fortran 95 files). Then compile main_template.f90 and link to the object files generated previously. Do the same for test_install.f90. TEST_INSTALL provides a simple test of the installation. Note that an implementation of MPI has to be installed to compile POLSYS_GLP. Since both polsys_glp.f90 and main_template.f90 contain MPI primitives, it is advisable to compile these two files with compilers provided by the given MPI implementation. For example, if MPICH (<http://www-unix.mcs.anl.gov/mpi/mpich>) is installed, use mpif90 for compilation and linking, since it already has all the necessary include and link paths resolved. To run the created executable file <program_name> on four processors of a Beowulf cluster, issue the following command:

```
mpirun -np 4 program_name
```

There must be at least two processors to obtain solutions. One processor is the master processor that distributes the homotopy paths and collects the zeros, and the remaining processors track the paths. For the sample input file INPUT.DAT, the results should match those in the sample output file OUTPUT.DAT, although the zeros may be listed in a different order due to compiler/machine differences.

For a Beowulf cluster, a sample PBS script file is:

```
#!/bin/csh
#PBS -N polsys_glp
#PBS -l nodes=4:ppn=2
#PBS -l walltime=10:00:00
#
# Edit the next line to reflect your working directory.
cd ~
#
mpirun -np 4 program_name >& glp.out
# End of example script.
```

The modules and external subroutines in polsys_glp.f90 and lapack_glp.f can be stored in module and object libraries and need not be recompiled. The subroutine TARGET_SYSTEM_USER defines the polynomial system and its Jacobian matrix, or a dummy subroutine, and must be supplied on every call to POLSYS_GLP. However, if the user does not wish to change TARGET_SYSTEM_USER, its object code can be stored in the aforementioned object library.

Inquiries should be directed to Hai-Jun Su (suh@uci.eng.edu), J. Michael McCarthy (jmmccart@uci.edu), Department of Mechanical and Aerospace Engineering, UCI, Irvine, CA 92697, (949) 824-8051; or Layne T. Watson (ltw@cs.vt.edu), Departments of Computer Science and Mathematics, VPI & SU, Blacksburg, VA 24061, (540) 231-7540.

A.3 Mathematica Notebook for Preparing INPUT.DAT

A Mathematica package `ExportGLP` is developed for preparing the file `INPUT.DAT`. Two functions are included, one is `CheckGLP` for checking validity of a given GLP structure and the other is `ExportGLP` for exporting the polynomial system and GLP structure to the file `INPUT.DAT`.

CheckGLP Usage:

```
CheckGLP[PolSys, Partition, SetDeg]
```

where “PolSys” is a list of polynomials, “Partition” is the GLP structure and “Set-Deg” is the degree associated with the GLP structure. `CheckGLP` returns `TRUE` if the given GLP start system is valid and `FALSE` otherwise.

ExportGLP Usage 1:

```
ExportGLP[PolSys, Partition, SetDeg]
```

where the parameters are the same as above. A file `INPUT.DAT` is generated.

ExportGLP Usage 2:

ExportGLP[PolSys]

This option will generate 1-homogeneous start system. "PolSys" is a list of polynomials as above. A file INPUT.DAT is generated.

Mathematica Notebook Source Code

```
(* :Title: Prepressing tools for POLSYS_GLP *)
(* :Author: Hai-Jun Su and J.Michael McCarthy
    Email: suh@eng.uci.edu, jmmccart@uci.edu
    Address: Laboratory for the Analysis and Synthesis of Spatial Movement
            University of California at Irvine
            Irvine, CA 92697
    Tel: (949)824-8051
    Web: http://synthetica.eng.uci.edu/
*)
(* :Version: Mathematica 5.0 *)
(* :Keywords:
polynomial homotopy solver, POLSYS_GLP, generalized linear product
*)
(* :History: Version 1.0 by Hai-Jun Su, 2004. *)

(* :Summary:
    This package is for generating INPUT.DAT file for POLSYS_GLP a polynomial homotopy
    solver developed at Univeristy of California, Irvine.
*)

BeginPackage["Homotopy`ExportGLP`"]
CheckGLP::usage = "CheckGLP[PolSys_?ListQ, Partition_?ListQ, SetDeg_?ListQ] check if the \
validity of the given partition."

ExportGLP::usage = "ExportGLP exports the polynomial system to a data file. \n
Warning: the calling notebook must have a name. \n
Possible Options: \n
[PolSys_?ListQ, Partition_?ListQ, SetDeg_?ListQ, FileName_?StringQ]: full options \n
[PolSys_?ListQ]: 1 homogenous start system, filename = INPUT.DAT \n
[PolSys_?ListQ, Partition_?ListQ]: All set degree are 1, filename = INPUT.DAT \n
[PolSys_?ListQ, Partition_?ListQ, SetDeg_?ListQ]: filename = INPUT.DAT.\n
[PolSys_?ListQ, Partition_?ListQ, FileName_?StringQ]: All set degree are 1. \n "
```

```

Begin["Private"]
(***** Interface for POLSYS_GLP *****)
CheckGLP[PolSys_?ListQ, Partition_?ListQ, SetDeg_?ListQ] := Module[{NN = Length[PolSys], \
StartSystem, ValidFlag},
  If[Length[Partition] ? NN || Length[SetDeg] ? NN,
    Print["\nLength[PolSys] != Length[Partition] or Length[PolSys] !=
Length[SetDeg]"];
    Return[False];
  ];
  ValidFlag = Table[CheckGLPOneEqn[PolSys[[i]], Partition[[i]], SetDeg[[i]]], {i, 1, NN}];
  Print[ValidFlag];
  If[MemberQ[ValidFlag, False],
    Return[False], Return[True]]
]
CheckGLPOneEqn[Pol_, Partition_?ListQ, SetDeg_?ListQ] := Module[{StartSystem,
MonListAll, MonListPol, PL, ValidFlag},
  If[Length[Partition] ? Length[SetDeg],
    Print["Length[PolSys] != Length[SetDeg]"];
    Return[False];
  ];
  StartSystem = 1;
  Do[
    Do[
      Coef = Table[Random[Real, {-1, 1}, 20], {k, Length[Partition[[i]]}];
      StartSystem = StartSystem*(Coef.Partition[[i]] - 1);
      , {j, 1, SetDeg[[i]]}
    , {i, 1, Length[Partition]}
    ];
  StartSystem = Expand[StartSystem];
  MonListAll = Table[StartSystem[[i]], {i, Length[StartSystem]} /. {_Real -> 1};
  PL = N[Expand[Pol], 20];
  MonListPol = Table[PL[[i]], {i, Length[PL]} /. {_Real -> 1, _Complex -> 1};
  ValidFlag = Table[MemberQ[MonListAll, MonListPol[[i]]], {i, Length[MonListPol]}];
  If[MemberQ[ValidFlag, False],
    Return[False], Return[True]]
]
ExportGLP[PolSys_?ListQ] := ExportGLP[PolSys, "INPUT.DAT"];
ExportGLP[PolSys_?ListQ, FileName_?StringQ] :=
  Module[{Partition, SetDeg, Vars, PL, maxDeg},
    Vars = Variables[PolSys];
    Partition = Table[{Vars}, {i, Length[PolSys]}];
    PL = Expand[PolSys];

```

```

maxDeg = Table[Max[Table[Sum[Exponent[PL[[k, i]],
Vars[[j]]], {j, 1, Length[Vars]}], {i, 1, Length[PL[[1]]}], {k, 1, Length[PL]}];
SetDeg = Table[{maxDeg[[i]]}, {i, 1, Length[Partition]}];
ExportGLP[PolSys, Partition, SetDeg, FileName];

ExportGLP[PolSys_?ListQ, Partition_?ListQ] := ExportGLP[PolSys, Partition, "INPUT.DAT"];
ExportGLP[PolSys_?ListQ, Partition_?ListQ, SetDeg_?ListQ] :=
    ExportGLP[PolSys, Partition, SetDeg, "INPUT.DAT"];
ExportGLP[PolSys_?ListQ, Partition_?ListQ, FileName_?StringQ] := Module[{SetDeg},
    SetDeg = Table[Table[1, {j, 1, Length[Partition[[i]]}], {i, 1, Length[Partition]}];
    Print[SetDeg];
    ExportGLP[PolSys, Partition, SetDeg, "INPUT.DAT"];]

ExportGLP[PolSys_?ListQ, Partition_?ListQ, SetDeg_?ListQ, FileName_?StringQ] :=
Module[{OldDir, outputFile, vars, NumEQ, SYSPARTITION, PROBLEM, TITLE, TRACKTOL, FINALTOL, SINGTOL,
SSPAR5, NUMRR, AllTerms, CoeffAllTerms, TotalDEGs, TOTDG, MAXT},
    OldDir = Directory[];
    CurrentDir = ToFileName[Extract["FileName" /.
        NotebookInformation[EvaluationNotebook[], {1}, FrontEnd`FileName]];
    SetDirectory[CurrentDir];
    outputFile = OpenWrite[FileName];
    vars = Variables[PolSys];
    NumEQ = Length[PolSys];
    PROBLEM = "&PROBLEM NEW_PROBLEM = .TRUE. \n";
    TITLE = "TITLE = ' This data file is generated by Mathematica and intended
        for use POLSYS_GLP.'\n";
    TRACKTOL = "TRACKTOL = 1.D-04\n";
    FINALTOL = "FINALTOL = 1.D-12\n";
    SINGTOL = "SINGTOL = 0.0\n";
    SSPAR5 = "SSPAR(5) = 1.D+00\n";
    NUMRR = 1;
    SYSPARTITION = "&SYSGLPSET ROOT_COUNT_ONLY = .FALSE.\n";
    AllTerms = Table[Table[PolSys[[i, j]], {j, Length[PolSys[[i]]}], {i, Length[PolSys]}];
    CoeffAllTerms = AllTerms /. Table[vars[[i]] -> 1, {i, Length[vars]}];
    TotalDEGs = Table[Max[Table[Sum[Exponent[AllTerms[[i, j]], vars[[k]], {k, Length[vars]}],
        {j, Length[AllTerms[[i]]}], {i, Length[AllTerms]}];
    TOTDG = Product[TotalDEGs[[i]], {i, Length[TotalDEGs]}];
    MAXT = Max[Table[Length[PolSys[[i]]], {i, Length[PolSys]}];

    WriteString[outputFile, PROBLEM];
    WriteString[outputFile, TITLE];
    WriteString[outputFile, "\n"];
    WriteString[outputFile, TRACKTOL];

```

```

WriteString[outputFile, FINALTOL];
WriteString[outputFile, SINGTOL];
WriteString[outputFile, SSPAR5];
WriteString[outputFile, "NUMRR = ", NUMRR, "\n"];
WriteString[outputFile, "N = ", NumEQ, "\n"];
WriteString[outputFile, "\n"];
Do[
  WriteString[outputFile, "NUM_TERMS(", i, ") = ",
    Length[AllTerms[[i]]], "\n"];
  Do[
    Do[WriteString[outputFile, "  DEG(", i, ", ", j, ", ", k,
      ") = ", Exponent[AllTerms[[i, j]], vars[[k]], "\n"],
      {k, Length[vars]}];
    WriteString[outputFile, "  COEF(", i, ", ", j, ") = (",
      FortranForm[Re[N[CoeffAllTerms[[i, j]]]], ", ",
      FortranForm[Im[N[CoeffAllTerms[[i, j]]]], ")", "\n"],
      {j, Length[AllTerms[[i]]]}];
    WriteString[outputFile, "\n"];
    , {i, Length[AllTerms]}];
  WriteString[outputFile, "/\n"];

WriteString[outputFile, SYSPARTITION];

Do[WriteString[outputFile, "P(", i, ") = '", Partition[[i]], "'\n"],
  {i, 1, Length[Partition]}];
Do[WriteString[outputFile, "DG(", i, ") = '", SetDeg[[i]], "'\n"],
  {i, 1, Length[SetDeg]}];

WriteString[outputFile, "\n"];
Do[WriteString[outputFile, "NUM_SETS(", i, ") = ",
  Length[Partition[[i]]], "\n"];
Do[WriteString[outputFile, "  NUM_INDICES(", i, ", ", j, ") = ",
  Length[Partition[[i, j]]], "  SET_DEG(", i, ", ", j, ") = ", SetDeg[[i, j]], "\n"];
Do[WriteString[outputFile, "  INDEX(", i, ", ", j, ", ", k, ") = ",
  (Position[vars, Partition[[i, j, k]]][[1, 1]]), "\n"],
  {k, Length[Partition[[i, j]]]}];
  , {j, Length[Partition[[i]]]}];
  , {i, 1, NumEQ}];
WriteString[outputFile, "/\n"];

Print["FILE = ", FileName, "\nNumber of Equations = ", NumEQ,
"\nNumber of Variables = ", Length[vars], " ", vars,
"\nTotal Degree = ", TOTDG,

```

```

    "\nMaximun Number of Terms = ", MAXT, "\n PARTITION = \n", Partition];

    Close[outputFile];
    SetDirectory[OldDir];
] (*End of Module*)
End[]
EndPackage[]

```

A.4 Sample INPUT.DAT and OUTPUT.DAT

Here we use the example of CS6 design problem in Table 5.4 to show the result of POLSYS_GLP. Interested readers can compare the solutions in the file OUTPUT.DAT with Table 5.5.

INPUT.DAT

```

&PROBLEM  NEW_PROBLEM = .TRUE.
TITLE = ' CS6 DESIGN PROBLEM: FIVE QUADRICS, NO SOLUTIONS AT INFINITY, 26 REAL SOLUTIONS.

TRACKTOL = 1.0D-04  FINALTOL = 1.0D-14  SINGTOL = 0.0D0  SSPAR(5) = 1.0D0
NUMRR = 1
N = 5
DEG=40000*0

NUM_TERMS(1) = 18
    DEG(1,1,1) = 0
    DEG(1,1,2) = 0
    DEG(1,1,3) = 0
    DEG(1,1,4) = 0
    DEG(1,1,5) = 0
COEF(1,1) = (2.93958190576569E-02, 0.000000000000000E+00)

```