



This document is part of LIBPF
Paolo Greppi - 3iP © Copyright 2004 - 2008
All rights reserved; do not distribute without permission.

LIBPF

Case study Polyols from Biomass

Summary

Introduction.....	2
Problem description.....	3
The components.....	3
Flowsheet.....	3
Process.....	4
Problem translation in LIBPF.....	7
Input.....	7
Flowsheet output.....	10
Resolution.....	10



Introduction

LIBPF (**LIB**rary for **P**rocess **F**lowsheeting) is a modelling tool to rapidly prototype and deploy small-footprint custom applications implementing computations for training, process engineer support, on-line process diagnostic, and data reconciliation. For more information regarding **LIBPF**, please see the website <http://www.libpf.com>.

The purpose of the **Polyols from Biomass LIBPF demo** is to illustrate the process flowsheet solving capabilities of LIBPF, plus the front-end (User Interface), database repository and calculation server, based on an actual process. The prototype shows basic functionality, but it could be easily extended to become a tool for off-line process diagnostic and reconciliation.

Bulk chemical production based on biomass feedstock is a promising alternative to the traditional petrochemical route. One interesting process is the conversion of short-chain carbohydrates to polyols via high pressure hydrogenation and cracking; this process is licensed by International Polyol Chemicals Inc. (**IPCI**, <http://www.polyolchem.com>) and has recently found the first industrial applications in Changchun (PRC) by Global Biochem Technology Group Co. Ltd (<http://www.globalbiochem.com/>).

The demo is based on a simplified steady-state model of the process, with illustrative purposes only; the model should not be considered an accurate representation of the proposed process. Informations on the IPCI Polyols from Biomass process were drawn from public literature sources and data from U.S. Patent No. 6479713 "*Hydrogenolysis of 5-carbon sugars, sugar alcohols, and other methods and compositions for reactions involving hydrogen*" by T.A.Werpy, J.G. Frye Jr., A.H.Zacher; D.J.Miller of Battelle Memorial Institute (U.S.A.).



Problem description

The components

To represent qualitatively the process, a selection of the numerous chemical species present has been made, which led to the following selection:

Nome	Formula	PM
H2O	CO2	18.0153
Saccarosio	C12H22O11	342.3
Glucosio	C6H12O6	180.158
Aldolo (sorbitolo)	C6H14O6	182.174
EG	C2H6O2	62.0684
Glicerina	C3H8O3	92.0947
PG	C3H8O2	76.0953
1,3-BDO	C4H10O2	90.1222
1,4-BDO	C4H10O2	90.1222
Etanolo	C2H6O	46.069
H2	H2	2.01588
CO2	CO2	44.0098

Flowsheet

The process flowsheet is based on the configuration termed “Type 3” by IPCI:

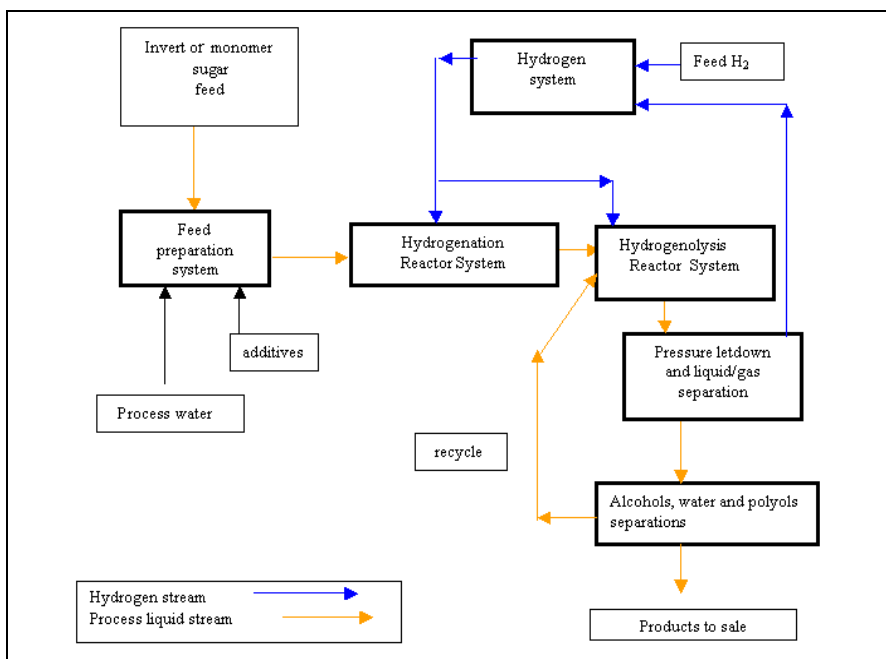


Figure 1 - Process scheme from IPCI web site



Process

Glucose in watery solution is converted to sorbitol in the hydrolysis reactor where it is contacted with a hydrogenation catalyst in the presence of gaseous hydrogen and under pressure.

Two reactions are assumed taking place in the hydrolysis reactor:

Nome	PM	coeff	C	H	O
Glucosio	180.158	-1	-6	-12	-6
Etanolo	46.069	2	4	12	2
CO2	44.0098	2	2	0	4
DELTA	-0.0004		0	0	0

Nome	PM	coeff	C	H	O
Glucosio	180.158	-1	-6	-12	-6
H2	2.01588	-1	0	-2	0
Aldolo (sorbitolo)	182.174	1	6	14	6
DELTA	0.00012		0	0	0

The conversion of glucose is assumed 99 % molar to sorbitol and 0.5 % molar to ethanol/CO₂.

The hydrolysis educt is mixed with an additional stream of hydrogen and a basic promoter, then fed to a hydrocracking reactor where the sorbitol is split into smaller chain polyols such as glycerine, ethylenglicol, propylenglicol etc.

The reaction scheme of the hydrocracking according to the patent is shown in figure 2.

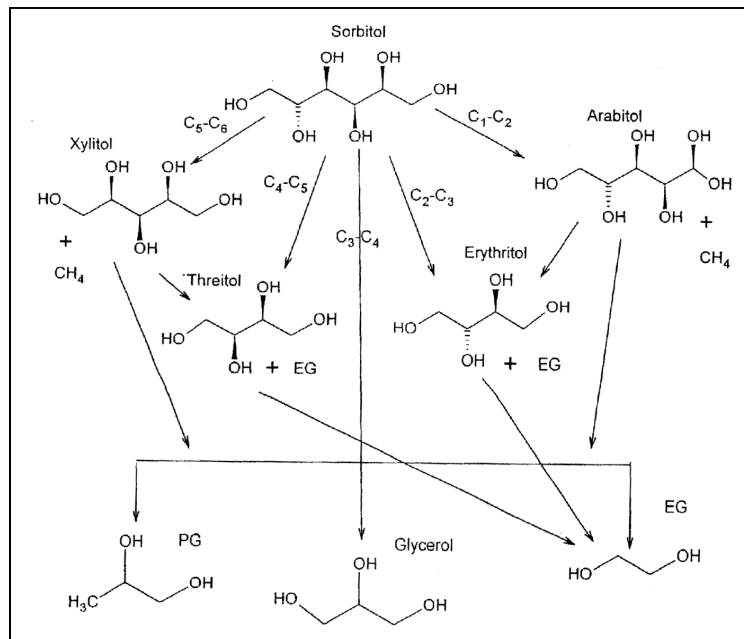


Figure 2 - Hydrocracking reaction scheme

The assumed reaction scheme is based on the following five reactions:

Nome	PM	coeff	C	H	O
Aldolo (sorbitolo)	182.174	-1	-6	-14	-6
H2	2.01588	-2	0	-4	0
EG	62.0684	3	6	18	6
DELTA	-0.00056		0	0	0

Nome	PM	coeff	C	H	O
Aldolo (sorbitolo)	182.174	-1	-6	-14	-6
H2	2.01588	-3	0	-6	0
PG	76.0953	2	6	16	4
H2O	18.0153	2	0	4	2
DELTA	-0.00044		0	0	0

Nome	PM	coeff	C	H	O
Aldolo (sorbitolo)	182.174	-1	-6	-14	-6
H2	2.01588	-1	0	-2	0
Glicerina	92.0947	2	6	16	6
DELTA	-0.00048		0	0	0



Nome	PM	coeff	C	H	O
Aldolo (sorbitolo)	182.174	-1	-6	-14	-6
H2	2.01588	-3.5	0	-7	0
1,3-BDO	90.1222	1.5	6	15	3
H2O	18.0153	3	0	6	3
DELTA	-0.00038		0	0	0

Nome	PM	coeff	C	H	O
Aldolo (sorbitolo)	182.174	-1	-6	-14	-6
H2	2.01588	-5	0	-10	0
Etanolo	46.069	3	6	18	3
H2O	18.0153	3	0	6	3
DELTA	-0.0005		0	0	0

The single pass conversion of sorbitol in this reactor has been assumed of 85 %. The molar selectivity has been based on an assumed Carbon molar selectivity (CMS) as follows:

Prodotto	coeff	C	coeff*C	Carbon molar selectivity	Selettività rispetto al
EG	3	2	6	25.0%	28.3%
PG	2	3	6	35.0%	26.4%
BDO	1.5	4	6	10.0%	5.7%
Glicerina	2	3	6	25.0%	18.9%
Etanolo	3	2	6	5.0%	5.7%
TOTALE				100.0%	85.0%

A simple black-box separation is assumed, taking into account partial recovery of unreacted sorbitol which is recycled to the hydrocracking reactor to improve upon the single-pass conversion.

The separation recoveries to the recycle were:

- 98 % for sorbitol
- 15 % for water
- 5 % for the impurities (including polyols).



Problem translation in LIBPF

Input

The problem is specified in LIBPF in four steps:

- 1) subclass a new class from `flowsheet` to represent the case

```
class ipci : public flowsheet<zero_zero> {
private:
    static std::string type_;
    void maketables(long);
public:
    ipci(const std::string &t, const std::string &d, modelBase *p);
    ipci(long cid);
    void makeuserassembly(std::list<assignment *>::iterator &p);
    void setup(void);
    const std::string &type(void) const { return type_; }
}; // ipci
```

- 2) define case connectivity in `maketables` virtual method of new type

```
void ipci::maketables(long CID) {
    diagnostic(2, "entered for " << TAG);
    // Non embedded objects
    if (CID == -1) {
        diagnostic(2, "Define unit operations");

        makeVertex<genflashN1> ("RXHYL", "Reattore di idrolisi", CID,
            boost::assign::map_list_of<std::string, long>("nReactions", 2),
            boost::assign::map_list_of<std::string, std::string>("embeddedTypeReactions[00]",
                "reactionYield")("embeddedTypeReactions[01]", "reactionYield"));
        makeVertex<genflashN1> ("RXHYC", "Reattore di idrocracking", CID,
            boost::assign::map_list_of<std::string, long>("nReactions", 5),
            boost::assign::map_list_of<std::string,
                std::string>("embeddedTypeReactions[00]", "reactionYield") ("embeddedTypeReactions[01]",
                "reactionYield")("embeddedTypeReactions[02]", "reactionYield")("embeddedTypeReactions[03]",
                "reactionYield")("embeddedTypeReactions[04]", "reactionYield"));
        makeVertex<gensep2> ("SEP", "Recupero sorbitolo", CID);

        diagnostic(2, "Define stream objects and connect");
        makeEdge<streamVL> ("SYRUP", "Sciroppo di glucosio", "source", "out", "RXHYL", "in", CID);
        makeEdge<streamVL> ("H2L", "Idrogeno ad idrogenolisi", "source", "out", "RXHYL", "in",
            CID);
        makeEdge<streamVL> ("KAT", "Catalizzatore", "source", "out", "RXHYL", "in", CID);
        makeEdge<streamVL> ("RXHYLOUT", "Miscela da idrolisi", "RXHYL", "out", "RXHYC", "in", CID);
        makeEdge<streamVL> ("PRO", "Promotore basico", "source", "out", "RXHYC", "in", CID);
        makeEdge<streamVL> ("H2C", "Idrogeno ad idrocracking", "source", "out", "RXHYC", "in",
            CID);
        makeEdge<streamVL> ("RXHYCOUT", "Miscela da idrocracking", "RXHYC", "out", "SEP", "in",
            CID);
        makeEdge<streamVL> ("RECY", "Sorbitolo recuperato", "SEP", "out1", "RXHYC", "in", CID);
        makeEdge<streamVL> ("PRODUCT", "Prodotti", "SEP", "out2", "sink", "in", CID);
    }
} // ipci::maketables
```

- 3) set up model inputs in `setup` virtual method of new type



```

void ipci::setup(void) {
  diagnostic(2, " entered for " << TAG);
  O("SYRUP")->S("flowoption")->set("Mw");
  dynamic_cast<stream *>(O("SYRUP"))->clearcomposition();
  O("SYRUP")->O("Tphase")->Q("mdot")->set(10000.0, "kg/h");
  O("SYRUP")->O("Tphase")->Q("w", "H2O")->set(0.75);
  O("SYRUP")->O("Tphase")->Q("w", "Glucosio")->set(0.25);

  O("H2L")->S("flowoption")->set("Nx");
  dynamic_cast<stream *>(O("H2L"))->clearcomposition();
  O("H2L")->O("Tphase")->Q("ndot")->set(400.0, "Nm3/h");
  O("H2L")->O("Tphase")->Q("x", "H2")->set(1.0);

  O("KAT")->S("flowoption")->set("Mw");
  dynamic_cast<stream *>(O("KAT"))->clearcomposition();
  O("KAT")->O("Tphase")->Q("mdot")->set(100.0, "kg/h");
  O("KAT")->O("Tphase")->Q("w", "H2O")->set(0.99);
  O("KAT")->O("Tphase")->Q("w", "Catalizzatore")->set(0.01);

  O("PRO")->S("flowoption")->set("Mw");
  dynamic_cast<stream *>(O("PRO"))->clearcomposition();
  O("PRO")->O("Tphase")->Q("mdot")->set(500.0, "kg/h");
  O("PRO")->O("Tphase")->Q("w", "H2O")->set(0.);
  O("PRO")->O("Tphase")->Q("w", "NaOH")->set(1.0);

  O("H2C")->S("flowoption")->set("Nx");
  dynamic_cast<stream *>(O("H2C"))->clearcomposition();
  O("H2C")->O("Tphase")->Q("ndot")->set(1000.0, "Nm3/h");
  O("H2C")->O("Tphase")->Q("x", "H2")->set(1.0);

  O("RXHYL")->Q("T")->set(150+273.15, "K");
  O("RXHYL")->Q("P")->set(10, "bar");
  O("RXHYL")->O("reactions", 0)->Q("z")->set(0.99);
  O("RXHYL")->O("reactions", 0)->Q("coeff", "Glucosio")->set(-1.0);
  O("RXHYL")->O("reactions", 0)->Q("coeff", "H2")->set(-1.0);
  O("RXHYL")->O("reactions", 0)->Q("coeff", "Sorbitolo")->set(1.0);
  O("RXHYL")->O("reactions", 0)->I("keycomp")->set(components.lookup("Glucosio"));
  O("RXHYL")->O("reactions", 1)->Q("z")->set(0.5); // 50 % dell' 1% residuo
  O("RXHYL")->O("reactions", 1)->Q("coeff", "Glucosio")->set(-1.0);
  O("RXHYL")->O("reactions", 1)->Q("coeff", "Etanolo")->set(2.0);
  O("RXHYL")->O("reactions", 1)->Q("coeff", "CO2")->set(2.0);
  O("RXHYL")->O("reactions", 1)->I("keycomp")->set(components.lookup("Glucosio"));

  O("RXHYC")->Q("T")->set(150+273.15, "K");
  O("RXHYC")->Q("P")->set(80, "bar");
  O("RXHYC")->O("reactions", 0)->Q("z")->set(0.283);
  O("RXHYC")->O("reactions", 0)->Q("coeff", "Sorbitolo")->set(-1.0);
  O("RXHYC")->O("reactions", 0)->Q("coeff", "H2")->set(-2.0);
  O("RXHYC")->O("reactions", 0)->Q("coeff", "EG")->set(3.0);
  O("RXHYC")->O("reactions", 0)->I("keycomp")->set(components.lookup("Sorbitolo"));
  O("RXHYC")->O("reactions", 1)->Q("z")->set(0.369);
  O("RXHYC")->O("reactions", 1)->Q("coeff", "Sorbitolo")->set(-1.0);
  O("RXHYC")->O("reactions", 1)->Q("coeff", "H2")->set(-3.0);
  O("RXHYC")->O("reactions", 1)->Q("coeff", "PG")->set(2.0);
  O("RXHYC")->O("reactions", 1)->Q("coeff", "H2O")->set(2.0);
  O("RXHYC")->O("reactions", 1)->I("keycomp")->set(components.lookup("Sorbitolo"));
  O("RXHYC")->O("reactions", 2)->Q("z")->set(0.125);
  O("RXHYC")->O("reactions", 2)->Q("coeff", "Sorbitolo")->set(-1.0);
  O("RXHYC")->O("reactions", 2)->Q("coeff", "H2")->set(-3.5);
  O("RXHYC")->O("reactions", 2)->Q("coeff", "BD014")->set(1.5);

```



```

0("RXHYC")->0("reactions", 2)->Q("coeff", "H2O")->set(3.0);
0("RXHYC")->0("reactions", 2)->I("keycomp")->set(components.lookup("Sorbitolo"));
0("RXHYC")->0("reactions", 3)->Q("z")->set(0.478);
0("RXHYC")->0("reactions", 3)->Q("coeff", "Sorbitolo")->set(-1.0);
0("RXHYC")->0("reactions", 3)->Q("coeff", "H2")->set(-1.0);
0("RXHYC")->0("reactions", 3)->Q("coeff", "Glicerina")->set(2.0);
0("RXHYC")->0("reactions", 3)->I("keycomp")->set(components.lookup("Sorbitolo"));
0("RXHYC")->0("reactions", 4)->Q("z")->set(0.274);
0("RXHYC")->0("reactions", 4)->Q("coeff", "Sorbitolo")->set(-1.0);
0("RXHYC")->0("reactions", 4)->Q("coeff", "H2")->set(-5.0);
0("RXHYC")->0("reactions", 4)->Q("coeff", "Etanolo")->set(3.0);
0("RXHYC")->0("reactions", 4)->Q("coeff", "H2O")->set(3.0);
0("RXHYC")->0("reactions", 4)->I("keycomp")->set(components.lookup("Sorbitolo"));

0("SEP")->Q("T")->set(150+273.15, "K");
0("SEP")->Q("P")->set(80, "bar");
0("SEP")->Q("outSplit[0]", "H2O")->set(0.15);
0("SEP")->Q("outSplit[0]", "NaOH")->set(0.15);
0("SEP")->Q("outSplit[0]", "Catalizzatore")->set(0.05);
0("SEP")->Q("outSplit[0]", "Glucosio")->set(0.05);
0("SEP")->Q("outSplit[0]", "Sorbitolo")->set(0.98);
0("SEP")->Q("outSplit[0]", "EG")->set(0.05);
0("SEP")->Q("outSplit[0]", "Glicerina")->set(0.05);
0("SEP")->Q("outSplit[0]", "PG")->set(0.05);
0("SEP")->Q("outSplit[0]", "BD013")->set(0.05);
0("SEP")->Q("outSplit[0]", "BD014")->set(0.05);
0("SEP")->Q("outSplit[0]", "Etanolo")->set(0.05);
0("SEP")->Q("outSplit[0]", "H2")->set(0.0);
0("SEP")->Q("outSplit[0]", "CO2")->set(0.0);

diagnostic(3, "Making selected outputs visible in GUI");
0("RXHYC")->Q("duty")->setOutput();
0("RXHYC")->Q("alfa")->setOutput();
0("RXHYL")->Q("duty")->setOutput();
0("RXHYL")->Q("alfa")->setOutput();
0("RXHYLOUT")->0("Tphase")->Q("mdotcomps", "EG")->setOutput();

diagnostic(3, "Provide initial estimate for streams which will be cut");
0("RECY")->Q("T")->set(150+273.15, "K");
0("RECY")->Q("P")->set(80, "bar");
0("RECY")->S("flowoption")->set("Mw");
dynamic_cast<stream *>(0("RECY"))->clearcomposition();
0("RECY")->0("Tphase")->Q("mdot")->set(2000.0, "kg/h");
0("RECY")->0("Tphase")->Q("w", "H2O")->set(0.8);
0("RECY")->0("Tphase")->Q("w", "Sorbitolo")->set(0.2);

diagnostic(3, "Defining cut streams");
addcut("RECY");
} // ipci::setup

```

- 4) enter feedback specifications to be converged iteratively in makeuserassembly virtual method of new type

```

void ipci::makeuserassembly(std::list<assignment *>::iterator &p) {
    long int i(0);
    MakeAssignment(*0("SEP")->Q("outSplit[0]", "H2O"), *0("SEP")->Q("outSplit[0]", "H2O") * (1.0
+ (0.77 - *0("RECY")->0("Tphase")->Q("w", "H2O")) / *0("RECY")->0("Tphase")->Q("w", "H2O")),
    "Porta la percentuale d'acqua nel riciclo all'80%");
} // ipci::makeuserassembly

```

Flowsheet output

The connectivity is represented within LIBPF as the Directed Cyclic Graph shown in figure 3:

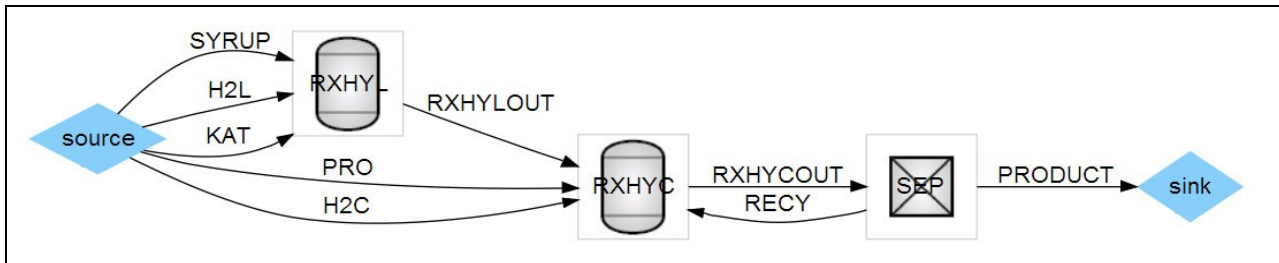


Figure 3 - Internal representation of flowsheet

Resolution

The command:

```
addcut("RECY");
```

in the setup method cuts the RECY and makes the graph a Directed Acyclic Graph shown in figure 4.

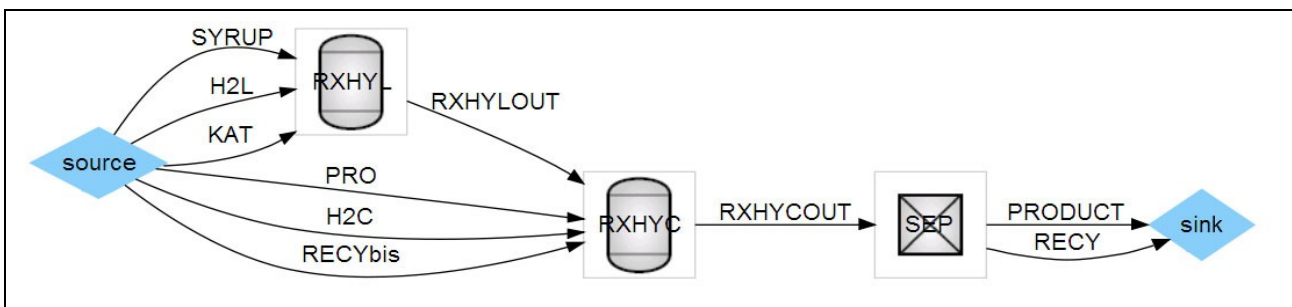


Figure 4 - Internal representation of flowsheet after cutting streams

The new stream RECYbis becomes a feedstream, whose values are initialized appropriately in the setup method; the DAG can be then solved sequentially.

For maximum speed only two sequential iterations are performed, then the solution proceeds simultaneously. The timings in typical modern hardware are 5 s for the initial resolution and 2 s for the subsequent resolutions.