The Y A_k Ψ representation

This converts the stoichiometric representation dx/dt = S v(x) of a chemical
reaction network to the form
S v(x) = Y A_k Ψ(x) = Y G K Ψ(x).

Here A_k is the Laplacian of the complexes graph and Y is the complexes to species matrix. The factoring A_k = G K is found in
V. Katsnelson's UCSD undergrad honors thesis.
It also computes properties of A_k and the deficiency of the network.

In[12]:=
```
S = Transpose[{{-1, 1, 0, 0, 1, 1}, {-1, 1, 0, 0, 0, 0}, {-1, 0, 1, 1, 0, 0}, {-1, 1, -1, 1, 0, 0},
    {0, 0, -1, 1, -1, 0}, {1, -1, 0, 0, 0, 0}, {0, 0, 1, -1, 0, 0}}](*stoichiometric matrix*);
S //
 MatrixForm
```

Out[13]//MatrixForm=

$$\begin{pmatrix} -1 & -1 & -1 & -1 & 0 & 1 & 0 \\ 1 & 1 & 0 & 1 & 0 & -1 & 0 \\ 0 & 0 & 1 & -1 & -1 & 0 & 1 \\ 0 & 0 & 1 & 1 & 1 & 0 & -1 \\ 1 & 0 & 0 & 0 & -1 & 0 & 0 \\ 1 & 0 & 0 & 0 & 0 & 0 & 0 \end{pmatrix}$$

In[14]:=
```
(*mr3:Function returning the rank of matrix
  m.For compatibility with earlier versions of Mathematica*)
mr3[m_] := Length[Select[RowReduce[m], ! (Dot[#, #] === 0) &]];

(*uUnion performs an unsorted union*)
uUnion[x_] := Reap[Sow[1, x], _, #1 &][[2]]
```

In[16]:=
```
outOf1Make2[vec_] := {Map[ Max[0, #] &, vec], - Map[ Min[0, #] &, vec]}
```

■ ** Computation of Y **

In[17]:=
```
(* matrixY takes the Stoichiometric matrix as intput and yields a
list of three matrices.
The first is matrix Y, the second is a list of the input complexes for all
the reactions (with repetitions), and the third is a list of the output
complexes for all the reactions (with repetitions). *)
```

In[18]:=
```
matrixY[Smat_] :=
 Module[{transposeS = Transpose[Smat], InputMatrix, OutputMatrix, ActualY}, InputMatrix = {};
  OutputMatrix = {};
  Y = {};
  Y = Flatten[Map[outOf1Make2[#] &, transposeS], 1];
  InputMatrix = Map[outOf1Make2[#][[2]] &, transposeS];
  OutputMatrix = Map[outOf1Make2[#][[1]] &, transposeS];
  ActualY = Transpose[uUnion[Y]];
  {ActualY, InputMatrix, OutputMatrix}]
```

In[19]:=
```
Y = matrixY[S];
```

In[20]:=
```
(* Here is the matrix Y. *)
```

In[21]:=
```
MatrixForm[Y[[1]]]
```

Out[21]//MatrixForm=

$$\begin{pmatrix} 0 & 1 & 0 & 0 & 0 & 1 & 0 & 0 & 0 \\ 1 & 0 & 1 & 0 & 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 & 1 & 0 & 1 & 1 \\ 0 & 0 & 0 & 1 & 1 & 0 & 1 & 0 & 0 \\ 1 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 \\ 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \end{pmatrix}$$

In[22]:=
```
(* Here are the input complexes. They may be identified with
monomials whose exponents are given by the rows of this matrix. *)
```

In[23]:=
```
MatrixForm[Transpose[Y[[2]]]]
```

Out[23]//MatrixForm=

$$\begin{pmatrix} 1 & 1 & 1 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 1 \\ 0 & 0 & 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 \end{pmatrix}$$

In[24]:=
```
(* Here are the output complexes. *)
```

In[25]:=
```
MatrixForm[Transpose[Y[[3]]]]
```

Out[25]//MatrixForm=

$$\begin{pmatrix} 0 & 0 & 0 & 0 & 0 & 1 & 0 \\ 1 & 1 & 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 & 0 & 1 \\ 0 & 0 & 1 & 1 & 1 & 0 & 0 \\ 1 & 0 & 0 & 0 & 0 & 0 & 0 \\ 1 & 0 & 0 & 0 & 0 & 0 & 0 \end{pmatrix}$$

** Computation of G **

In[26]:=
```
(* matrixG takes in matrix Y, the input complexes matrix,
and the output complexes matrix, and yields matrix G. *)
```

In[27]:=
```
matrixG[{Ymat_, Inputmatrix_, Outputmatrix_}] :=
 Module[{ActualY, InputMatrix, OutputMatrix, ActualK, autvec},
  ActualY = Ymat;
  InputMatrix = Inputmatrix;
  OutputMatrix = Outputmatrix;
  ActualK = {};
  For[i = 1, i ≤ Length[InputMatrix], i++,
   (* We will have Y.autvec =
    i'th column of S. InputMatrix[[i]] is the input complex of the i'th reaction. If the j'
     th column of matrix Y is InputMatrix[[i]], then the j'th entry of autvec will be -1.
   *)
   autvec = Table[If[j == Flatten[Position[Transpose[ActualY], InputMatrix[[i]]]][[1]], -1,
      (* OutputMatrix[[i]] is the output complex of the i'
       th reaction. If the j'th column of matrix Y is OutputMatrix[[i]],
      then the j'th entry of autvec will be 1. Otherwise, it will be 0.
      *)
      If[j == Flatten[Position[Transpose[ActualY], OutputMatrix[[i]]]][[1]], 1, 0]],
     {j, Length[ActualY[[1]]]}];
   (* ActualK will be the matrix consisting
    of all the autvec's generated at each iteration in our For loop.
   *)
   ActualK = Append[ActualK, autvec];
  ];
  Transpose[ActualK]]
```

In[28]:=
```
G = matrixG[Y];
G//MatrixForm
```

Out[29]//MatrixForm=

$$\begin{pmatrix} 1 & 0 & 0 & 0 & 0 & 0 & 0 \\ -1 & -1 & -1 & 0 & 0 & 1 & 0 \\ 0 & 1 & 0 & 0 & 0 & -1 & 0 \\ 0 & 0 & 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & -1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & 0 & -1 \\ 0 & 0 & 0 & 0 & -1 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 1 \end{pmatrix}$$

(* Check to make sure YG=S *)

In[30]:=
```
Print["YG = ", MatrixForm[Y[[1]].G], ",  S=" , MatrixForm[S]];
Y[[1]].G == S
```

$$YG = \begin{pmatrix} -1 & -1 & -1 & -1 & 0 & 1 & 0 \\ 1 & 1 & 0 & 1 & 0 & -1 & 0 \\ 0 & 0 & 1 & -1 & -1 & 0 & 1 \\ 0 & 0 & 1 & 1 & 1 & 0 & -1 \\ 1 & 0 & 0 & 0 & -1 & 0 & 0 \\ 1 & 0 & 0 & 0 & 0 & 0 & 0 \end{pmatrix}, \quad S = \begin{pmatrix} -1 & -1 & -1 & -1 & 0 & 1 & 0 \\ 1 & 1 & 0 & 1 & 0 & -1 & 0 \\ 0 & 0 & 1 & -1 & -1 & 0 & 1 \\ 0 & 0 & 1 & 1 & 1 & 0 & -1 \\ 1 & 0 & 0 & 0 & -1 & 0 & 0 \\ 1 & 0 & 0 & 0 & 0 & 0 & 0 \end{pmatrix}$$

Out[31]= True

** Computing Ψ (x) **

In[32]:=
```
makeMonomial[Ymat_, symb_] :=
  Module[{tmp, lista, pow, mat}, mat = Refine[Sign[Ymat], Map[# > 0 &, Variables[-Ymat]]];
   pow = Table[0, {Length[mat[[1]]]}];
   tmp = Table[If[mat[[i, j]] ≥ 0, 1, If[pow[[j]] > 0, symb[i] (-Ymat[[i, j]]), pow[[j]] ++;
       symb[i] ^ (-Ymat[[i, j]])]], {i, 1, Length[mat]}, {j, 1, Length[mat[[1]]]}];
   lista = Map[Apply[Times, #] &, Transpose[tmp]]; lista];
```

In[33]:=
```
Psi = makeMonomial[-Y[[1]], x];
Print["Ψ(x)= ", MatrixForm[Psi]]
```

$$\Psi(x) = \begin{pmatrix} x[2]\, x[5]\, x[6] \\ x[1] \\ x[2] \\ x[3]\, x[4] \\ x[2]\, x[4] \\ x[1]\, x[3] \\ x[4] \\ x[3]\, x[5] \\ x[3] \end{pmatrix}$$

** Computation of K **

In[35]:=
```
(* matrixK takes matrix G as input and yields matrix K. *)
```

In[36]:=
```
matrixK[Gmat_] := Module[{G, ActualK}, Clear[k];
    G = {};
    ActualK = Transpose[Gmat];
    n = 1;
(* K is a #(reactions) x #(complexes) matrix. For each complex...
*)
    For[i = 1, i <= Length[Transpose[ActualK]], i++,
(* If complex i participates in reaction j, then the j'th entry of tvec
is the rate constant k[i,j]. It's 0 otherwise.
*)
  tvec = Table[If[Transpose[ActualK][[i]][[j]] == -1, position =
    Flatten[Position[ActualK[[j]], 1]][[1]];
        k[i, position], 0], {j, 1, Length[ActualK]}];
  G = Append[G, tvec];
  ];
 Transpose[G]
]
```

In[37]:=
```
K = matrixK[G];
K//MatrixForm
```

Out[38]//MatrixForm=

$$\begin{pmatrix} 0 & k[2,1] & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & k[2,3] & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & k[2,4] & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & k[6,5] & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & k[8,7] & 0 \\ 0 & 0 & k[3,2] & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & k[7,9] & 0 & 0 \end{pmatrix}$$

** Computation of A_K **

In[39]:=
```
Ak = G.K;
Ak // MatrixForm
```

Out[40]//MatrixForm=

$$\begin{pmatrix} 0 & k[2,1] & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & -k[2,1]-k[2,3]-k[2,4] & k[3,2] & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & k[2,3] & -k[3,2] & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & k[2,4] & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & k[6,5] & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & -k[6,5] & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & -k[7,9] & k[8,7] & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & -k[8,7] & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & k[7,9] & 0 & 0 \end{pmatrix}$$

** We finally obtain our desired formula **

In[41]:=
```
Print["YGKΨ(x)= ", MatrixForm[Y[[1]]], MatrixForm[G], MatrixForm[K], MatrixForm[Psi]]
```

$$\mathrm{YGK\Psi(x)} = \begin{pmatrix} 0 & 1 & 0 & 0 & 0 & 1 & 0 & 0 & 0 \\ 1 & 0 & 1 & 0 & 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 & 1 & 0 & 1 & 1 \\ 0 & 0 & 0 & 1 & 1 & 0 & 1 & 0 & 0 \\ 1 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 \\ 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \end{pmatrix} \begin{pmatrix} 1 & 0 & 0 & 0 & 0 & 0 & 0 \\ -1 & -1 & -1 & 0 & 0 & 1 & 0 \\ 0 & 1 & 0 & 0 & 0 & -1 & 0 \\ 0 & 0 & 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & -1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & 0 & -1 \\ 0 & 0 & 0 & 0 & -1 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 1 \end{pmatrix}$$

$$\begin{pmatrix} 0 & k[2,1] & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & k[2,3] & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & k[2,4] & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & k[6,5] & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & k[8,7] & 0 \\ 0 & 0 & k[3,2] & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & k[7,9] & 0 & 0 \end{pmatrix} \begin{pmatrix} x[2]\,x[5]\,x[6] \\ x[1] \\ x[2] \\ x[3]\,x[4] \\ x[2]\,x[4] \\ x[1]\,x[3] \\ x[4] \\ x[3]\,x[5] \\ x[3] \end{pmatrix}$$

In[42]:=
```
(* The desired decomposition Y Ak Ψ(x). *)
```

In[43]:=
```
decomposition[S_] :=
 Module[ {Y, G, K, Ak, psi}, Y = matrixY[S]; G = matrixG[Y]; K = matrixK[G];
  Ak = G.K; psi = makeMonomial[-Y[[1]], x]; {Y[[1]], Ak, psi}]
```

In[44]:=
```
decomposition[S]
```

Out[44]=
```
{{{0, 1, 0, 0, 0, 1, 0, 0, 0}, {1, 0, 1, 0, 1, 0, 0, 0, 0}, {0, 0, 0, 1, 0, 1, 0, 1, 1},
  {0, 0, 0, 1, 1, 0, 1, 0, 0}, {1, 0, 0, 0, 0, 0, 0, 1, 0}, {1, 0, 0, 0, 0, 0, 0, 0, 0}},
 {{0, k[2, 1], 0, 0, 0, 0, 0, 0, 0}, {0, -k[2, 1] - k[2, 3] - k[2, 4], k[3, 2], 0, 0, 0, 0, 0, 0},
  {0, k[2, 3], -k[3, 2], 0, 0, 0, 0, 0, 0},
  {0, k[2, 4], 0, 0, 0, 0, 0, 0, 0}, {0, 0, 0, 0, 0, k[6, 5], 0, 0, 0},
  {0, 0, 0, 0, 0, -k[6, 5], 0, 0, 0}, {0, 0, 0, 0, 0, 0, -k[7, 9], k[8, 7], 0},
  {0, 0, 0, 0, 0, 0, 0, -k[8, 7], 0}, {0, 0, 0, 0, 0, 0, k[7, 9], 0, 0}},
 {x[2] x[5] x[6], x[1], x[2], x[3] x[4], x[2] x[4], x[1] x[3], x[4], x[3] x[5], x[3]}}
```

In[45]:=
```
(* linkageClasses takes the Laplacian of a graph (e.g. the A_k matrix in the Y A_k Ψ(x))
 as an input. It returns a vector with two components. The first
 component gives a list of all the linkage classes. For example,
{1,3,4} would be a linkage class where the first,
third and fourth complex participate in reactions with each other. By definition,
the linkage classes are disjoint.

   The second component in our output is a list of 3-
 tuples for each different complex in our chemical network. A 3-tuple will list
  all the reactions a certain complex particpates in. The first vector in the 3-
 tuple gives the index i of a particular complex y_i. The second vector in the 3-
 tuple gives indices for of each complex that y_i particates in a reaction with,
and where y_i is an input for that reaction.
   The third vector in the 3-
 tuple gives indices for of each complex that y_i particates in a reaction with,
and where y_i is an output for that reaction. For example,
the 3-tuple {{5},{1,2},{2,4,6}} means we have the reactions y_5→y_1,
y_5→y_2, y_2→y_5, y_4→y_5, and y_6→y_2. *)
```

In[46]:=
```
linkageClasses[Ak_] :=
 Module[{diag, A, newA, n, compgraph, classes, class, testvec, componentIn, componentOut},
  A = Ak;
  diag = Tr[A, List];
  newA = A - DiagonalMatrix[diag];
  n = Length[A[[1]]];
  compgraph = {};
  classes = {};
  For[i = 1, i ≤ n, i++,
   testvec = Table[If[j == i, 1, 0], {j, n}];
   componentIn = Flatten[Position[Map[ToString, newA.testvec], _?(# ≠ "0" &)]];
   componentOut = Flatten[Position[Map[ToString, testvec.newA], _?(# ≠ "0" &)]];
   compgraph = Append[compgraph, {{i}, componentIn, componentOut}];
   class = Union[componentIn, componentOut, {i}];
   classes = Append[classes, class];
  ];
  For[i = 1, i ≤ Length[classes], i++,
   For[j = i + 1, j ≤ Length[classes], j++,
     If[Intersection[classes[[i]], classes[[j]]] ≠ {},
       classes[[i]] = Union[classes[[i]], classes[[j]]]; classes = Delete[classes, j]; j = i];
    ];];
  {classes, compgraph}
 ]
```

In[47]:=
```
components = linkageClasses[Ak]
```

Out[47]=
```
{{{1, 2, 3, 4}, {5, 6}, {7, 8, 9}},
 {{{1}, {}, {2}}, {{2}, {1, 3, 4}, {3}}, {{3}, {2}, {2}}, {{4}, {}, {2}},
  {{5}, {}, {6}}, {{6}, {5}, {}}, {{7}, {9}, {8}}, {{8}, {7}, {}}, {{9}, {}, {7}}}}
```

In[48]:=
```
Print["The number of linkage classes is ", Length[components[[1]]]]
```

```
The number of linkage classes is 3
```

In[49]:=
```
(* The next formula gives the topological deficiency of a chemical reaction
 network. Its input is the stoichiometric matrix. *)
```

In[50]:=
```
topDeficiency[S_] := Module[ {y, ak, psi}, {y, ak, psi} = decomposition[S];
  Length[ ak ] - Length[ linkageClasses[ ak][[1]] ] - mr3[S]]
```

In[51]:=
```
topDeficiency[S]
```

Out[51]= 1

In[52]:=
```
(*The next formula gives the matrix deficiency of a chemical reaction
 network.Its input is the stoichiometric matrix.Two versions are given.A
 probabilistic version that runs on moderately big examples and a symbolic one.*)

matDeficiency[S_] := Module[{tmpak, y, ak, psi}, {y, ak, psi} = decomposition[S];
  Max[Table[tmpak = ak /. {Variables[ak] → RandomInteger[{1, 3}]};
    Length[NullSpace[y.tmpak]] - Length[NullSpace[tmpak]], {5}]]]

matDeficiencySymbolic[S_] := Module[{y, ak, psi}, {y, ak, psi} = decomposition[S];
  Length[NullSpace[y.ak]] - Length[NullSpace[ak]]]
```

In[54]:=
```
matDeficiency[S]
```

Out[54]= 1

In[55]:=
```
matDeficiencySymbolic[S]
```

Out[55]= 1