```
# MAPLE 11. Temp version, using new MABSys interface.
> infolevel[MABSys] := 0;
                              infolevel[MABSys] := 0

#########################################################################
# STEP 1 − DEFINITION OF THE CHEMICAL REACTION SYSTEM
#########################################################################

# The generalized chemical reaction system is defined using the NewReaction
# function of MABSys. The package cannot handle a system with a generic n.
# Thus one sets its value.

> n := 4;
                                       n := 4

# The kinetic constant for the polymerisation are k (for k+) and km (for k−)
# Observe that these reactions are defined as fast reactions.
# The functions NewReaction and NewMassActionLaw are parts of MABSys.

> RS := [
>     NewReaction(G+cat('P',n),H,NewMassActionLaw(alpha)),
>     NewReaction(H,G+cat('P',n),NewMassActionLaw(theta)),
>     NewReaction(G,G+M,NewMassActionLaw(rhof)),
>     NewReaction(H,H+M,NewMassActionLaw(rhob)),
>     NewReaction(M,M+P1,NewMassActionLaw(beta)),
>     NewReaction(M,0,NewMassActionLaw(deltaM)),
>     NewReaction(P1,0,NewMassActionLaw(deltaP)),
>     seq (NewReaction(cat('P',i)+P1,cat('P',i+1),
>                          NewMassActionLaw(cat('k_',i)),fast=true), i=1..n−1),
>     seq (NewReaction(cat('P',i+1),cat('P',i)+P1,
>                          NewMassActionLaw(cat('km_',i)),fast=true), i=1..n−1)
> ];

RS := [Reaction([G, P4], [H], MassActionLaw(alpha), false),

    Reaction([H], [G, P4], MassActionLaw(theta), false),

    Reaction([G], [G, M], MassActionLaw(rhof), false),

    Reaction([H], [H, M], MassActionLaw(rhob), false),

    Reaction([M], [M, P1], MassActionLaw(beta), false),

    Reaction([M], [], MassActionLaw(deltaM), false),

    Reaction([P1], [], MassActionLaw(deltaP), false),

    Reaction([2 P1], [P2], MassActionLaw(k_1), true),

    Reaction([P2, P1], [P3], MassActionLaw(k_2), true),

    Reaction([P3, P1], [P4], MassActionLaw(k_3), true),

    Reaction([P2], [2 P1], MassActionLaw(km_1), true),

    Reaction([P3], [P2, P1], MassActionLaw(km_2), true),

    Reaction([P4], [P3, P1], MassActionLaw(km_3), true)]

#########################################################################
#  STEP 2 − COMPUTATION OF THE RAW REDUCED MODEL (QSSA)
#########################################################################

# The list of the chemical species
> X := [G,H,M,seq(cat('P',n−i+1),i=1..n)];
                          X := [G, H, M, P4, P3, P2, P1]

# The ModelReduce algorithm is applied. It provides an enhanced
# implementation of the DifferentialModelReduction algorithm, based
# on the RegularChains package. It is part of MABSys.
```

```
> RawReducedModel := MABSys:−ModelReduce(RS,X,useConservationLaws=false) [1,1]:

# Some rewriting is performed over the raw reduced model
#     H(t) is removed and replaced by gamma0 − G(t)
#     The redundant ODE describing the evolution of H(t) is removed.
#     The ODEs describing the evolutions of P2(t), ..., Pn(t) are removed.
#     P1(t) is renamed as P(t).
#     The K_i parameters are introduced for legibility.

> RawReducedModel :=
>     subs
>       (H(t)=gamma0−G(t),
>        P1(t)=P(t),
>        seq(cat('k_',i)=cat('km_',i)*cat('K_',i)/cat('K_',i−1),i=1..n−1),
>        K_0=1,
>        [RawReducedModel[1],RawReducedModel[3],RawReducedModel[−1]]):
> RawReducedModel := map (simplify, RawReducedModel);

                     d                                                  4
RawReducedModel := [−− G(t) = theta gamma0 − theta G(t) − P(t)  K_3 alpha G(t),
                     dt

    d                                                                   d
    −− M(t) = rhof G(t) + rhob gamma0 − rhob G(t) − deltaM M(t), −− P(t) = − (
    dt                                                                  dt

    −4 theta gamma0 + 4 theta G(t) − beta M(t) + deltaP P(t)

             4                      /         3                   2
     + 4 P(t)  K_3 alpha G(t))  /  (16 P(t)  K_3 + 1 + 9 P(t)  K_2 + 4 K_1 P(t)
                               /

    )]

#########################################################################
# STEP 3 − COMPUTATION OF THE REDUCED MODEL (Lie Symmetries)
#########################################################################

# STEP 3.1
# Two parameters are removed from the raw reduced model.
# One restricts the search to Lie Symmetries of scaling type since
# they preserve the positivity property of the system parameters. It
# is allowed to perform a change of time.

> ParametersToRemove := [beta,deltaP]:
> UnchangedParameters := []:
> useTime := true:
> res := MABSys:−RemoveParameterByScalings
>       (RawReducedModel,ParametersToRemove, UnchangedParameters, useTime):
> IntermediateReducedModel := res[1];

IntermediateReducedModel := [

    d                                                 4
    −− G(t) = theta gamma0 − theta G(t) − P(t)  K_3 alpha G(t),
    dt

    d                                                                   d
    −− M(t) = rhof G(t) + rhob gamma0 − rhob G(t) − deltaM M(t), −− P(t) =
    dt                                                                  dt

                                                                   4
      −4 theta gamma0 + 4 theta G(t) − M(t) + P(t) + 4 P(t)  K_3 alpha G(t)
    − −−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−]
                          3                          2
                  16 P(t)  K_3 + 1 + 9 P(t)  K_2 + 4 K_1 P(t)

# The change of coordinates is automatically computed.
# The old variables are expressed as functions of the new ones.
```

```
> FirstChangeOfCoord := res[2];

                              t         deltaP M
FirstChangeOfCoord := [t = ------, M = --------, alpha = deltaP alpha,
                           deltaP        beta

                                      2                    2
                                deltaP  rhof         deltaP  rhob
    theta = deltaP theta, rhof = ------------, rhob = ------------,
                                     beta                 beta

    deltaM = deltaP deltaM]

# STEP 3.2
# Some parameters are managed to appear as factors in the right−hand sides
# of the ODE. This process somehow makes cylindric the algebraic variety
# defines the steady points.

# This step is not completely automatic since some ordering needs
# to be carefully chosen on the system variables and parameters.

> Pos := [  seq( cat('K_',n−i), i=1..n−1),
>           deltaM, theta, rhob, rhof, alpha, gamma0
>         ];

        Pos := [K_3, K_2, K_1, deltaM, theta, rhob, rhof, alpha, gamma0]

> PosOrNull := [G,M,P]:
> res := MABSys:−CylindrifySteadyPoints
>           (IntermediateReducedModel,Pos,PosOrNull,Triangularization=true):
> ReducedModel := res[1,1];

                 d                                  4
ReducedModel := [-- G(t) = (gamma0 − G(t) − P(t)  G(t)) theta,
                 dt

    d                                                        d
    -- M(t) = deltaM (rhof G(t) + gamma0 − G(t) − M(t)), -- P(t) =
    dt                                                       dt

                            3
                       alpha  (−M(t) + P(t))
    − ------------------------------------------------------
            3        3        2                          2
      16 P(t)  + alpha  + 9 P(t)  K_2 alpha + 4 K_1 P(t) alpha

                      3                                4
                 alpha  (−4 gamma0 + 4 G(t) + 4 P(t)  G(t))
     − ------------------------------------------------------------------]
               3        3         2                                  2
       (16 P(t)  + alpha  + 9 P(t)  K_2 alpha + 4 K_1 P(t) alpha ) rhob

> SecondChangeOfCoord := res[1,3];

                                  3          2
SecondChangeOfCoord := [K_3 = K_3 , K_2 = K_3  K_2, K_1 = K_3 K_1,

    rhob = deltaM theta rhob, rhof = deltaM theta rhob rhof,

                            4                   gamma0
    alpha = K_3 theta alpha , gamma0 = ---------------------,
                                       K_3 theta rhob alpha

                 G                     M              P
    G = ---------------------, M = ---------, P = ---------]
        K_3 theta rhob alpha       K_3 alpha      K_3 alpha

# Further change of variables for the legibility of the reduced system.
# The case n=1 is treated separatly.
```

```
> if n=1 then
>     ThirdChangeOfCoord := [rhob=theta*deltaM/rhob];
> else
>     ThirdChangeOfCoord := [rhob=1/rhob, alpha=1/alpha];
> end if:
> ReducedModel := MABSys:−ApplyChangeOfCoord(ReducedModel,ThirdChangeOfCoord);

                  d                                                4
ReducedModel := [−− G(t) = −theta (−gamma0 + G(t) + P(t)  G(t)),
                  dt

    d
    −− M(t) = deltaM (rhof G(t) + gamma0 − G(t) − M(t)),
    dt

                                                                                4
    d           −M(t) + P(t) − 4 rhob gamma0 + 4 rhob G(t) + 4 rhob P(t)  G(t)
    −− P(t) = − −−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−]
    dt                   3      3             2          2
                 16 P(t)  alpha  + 1 + 9 P(t)  K_2 alpha  + 4 K_1 P(t) alpha

# Here is the paper's formula:

> denominator :=
>     subs (K_0 = 1,
>       add ((i+1)^2*cat(K_,i)*P(t)^i*alpha^i, i=0..n−2)+
>         n^2*P(t)^(n−1)*alpha^(n−1)):

> PaperReducedModel :=
>     [ Diff (G(t),t) = theta*(gamma0 − G(t) − G(t)*P(t)^n),
>       Diff (M(t),t) = (rhof*G(t) − G(t) + gamma0 − M(t))*deltaM,
>       Diff (P(t),t) =
>       (n*rhob*(gamma0 − G(t) − G(t)*P(t)^n) + M(t) − P(t))/denominator
>     ];

                       d                                  4
PaperReducedModel := [−− G(t) = (gamma0 − G(t) − P(t)  G(t)) theta,
                       dt

    d
    −− M(t) = deltaM (rhof G(t) + gamma0 − G(t) − M(t)),
    dt

                                                 4
    d              4 rhob (gamma0 − G(t) − P(t)  G(t)) + M(t) − P(t)
    −− P(t) = −−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−]
    dt                3      3             2          2
              16 P(t)  alpha  + 1 + 9 P(t)  K_2 alpha  + 4 K_1 P(t) alpha

# The following command checks that the paper's formula is equal to
# the computed one. It should print a sequence of zeros.

> seq (simplify (rhs (PaperReducedModel [i]) − rhs (ReducedModel [i])),
>                                       i = 1 .. nops (ReducedModel));
                                    0, 0, 0

# The final change of coordinates given in the paper is the
# composition of the three ones computed above. The MABSys package
# provides functionnalities to compose changes of coordinates.

> FinalChangeOfCoord :=
>       MABSys:−ComposeChangeOfCoord
>           (FirstChangeOfCoord, SecondChangeOfCoord, ThirdChangeOfCoord);

                             t          deltaP alpha M           deltaP K_3 theta
FinalChangeOfCoord := [t = −−−−−−, M = −−−−−−−−−−−−−−, alpha = −−−−−−−−−−−−−−−,
                         deltaP          beta K_3                        4
                                                                 alpha
```

```
                                      2
                                  deltaP  deltaM theta rhof
    theta = deltaP theta, rhof = ------------------------,
                                         beta rhob

                 2
          deltaP  deltaM theta                                      3
    rhob = --------------------, deltaM = deltaP deltaM, K_3 = K_3 ,
                beta rhob

             2                                  rhob alpha gamma0      rhob alpha G
    K_2 = K_3  K_2, K_1 = K_3 K_1, gamma0 = ------------------, G = ------------,
                                                K_3 theta              K_3 theta

        alpha P
    P = -------]
          K_3
```

```
# The following command checks that the paper's change of coordinates
# maps the raw reduced model to the reduced model. It should print a
# sequence of zeros.

> ChangeOfCoordReducedModel :=
>     MABSys:−ApplyChangeOfCoord(RawReducedModel,FinalChangeOfCoord):
> seq (simplify (rhs(PaperReducedModel[i]) − rhs(ChangeOfCoordReducedModel[i])),
>                                      i = 1 .. nops (ReducedModel));
                                   0, 0, 0

######################################################################
# STEP 4 − QUALITATIVE ANALYSIS OF THE REDUCED MODEL
######################################################################

# SteadyPoint receives a Groebner basis of the ideal generated by the
# righthand side of the ODE. The ordering is chosen so that
#    − the leading monomials are plain variables
#    − the leading monomials are positive functions of the rhs variables.

> SteadyPoint := MABSys:−SteadyPointSystem(ReducedModel);

                                          4
SteadyPoint := [−theta (−gamma0 + G + P  G), deltaM (rhof G + gamma0 − G − M),

                                                          4
       −M + P − 4 rhob gamma0 + 4 rhob G + 4 rhob P  G
    − ------------------------------------------------]
          3      3          2          2
      16 P  alpha  + 1 + 9 P  K_2 alpha  + 4 K_1 P alpha

> SteadyPointOrder := plex (G, M, gamma0):
> SteadyPoint := Groebner:−Basis (SteadyPoint, SteadyPointOrder);

                          5                    4                          4
    SteadyPoint := [−P − P  + gamma0 (rhof + P ), M − P, −P + (rhof + P ) G]

> SteadyPoint :=
>     seq (Groebner:−LeadingMonomial (SteadyPoint [i], SteadyPointOrder) =
>       normal
>           (Groebner:−NormalForm
>               (Groebner:−LeadingMonomial (SteadyPoint [i], SteadyPointOrder),
>                SteadyPoint,
>                SteadyPointOrder)),
>         i = 1 .. nops (SteadyPoint));

                                      4
                                 (1 + P ) P                  P
          SteadyPoint := gamma0 = ----------, M = P, G = ---------
                                        4                       4
                                 rhof + P                rhof + P

# The Jacobian matrix of the system, evaluated at the steady point.
```

```
> J, Vars := ExpandedLiePointSymmetry:−JacobianMatrix (ReducedModel, [G,M,P]):
> J0 := map(normal, subs (SteadyPoint, J)):

# The denominator is denoted B. Observe that some of the system parameters
# (such as the K_1 and alpha) only occur in B.
# This permits us to consider B as a new free variable.

> J0 := subs (K_1 = solve (B − subs (P(t)=P,denominator), K_1), J0);
```

$$J0 := \begin{bmatrix} -\theta\,(1+P^4) & 0 & -\dfrac{4\,\theta\,P^4}{\rho_f + P^4} \\ \delta_M\,(\rho_f - 1) & -\delta_M & 0 \\ \dfrac{4\,\rho_b\,(1+P^4)}{B} & \dfrac{1}{B} & -\dfrac{P^4 + 16\,\rho_b\,P^4 + \rho_f}{B\,(\rho_f + P^4)} \end{bmatrix}$$

```
# Here is the paper’s formula for the Jacobian matrix.

> PaperJ0 :=
> <<−theta*(1+P^n)      | 0       | −n*theta*P^n/(P^n + rhof) >,
>  <(−1 + rhof)*deltaM | −deltaM | 0 >,
>  <−n*rhob*(1+P^n)/B | 1/B     | −(P^n+n^2*rhob*P^n+rhof)/(B*(P^n + rhof)) >
> >:

# If the paper’s formula is correct then the next command should produce
# the zero matrix.

> J0 − PaperJ0;
```

$$\begin{bmatrix} 0 & 0 & 0 \\ 0 & 0 & 0 \\ 0 & 0 & 0 \end{bmatrix}$$

```
# We now work on the paper’s formula with a symbolic exponent n.

> n := ’n’:

> J0 :=
> <<−theta*(1+P^n)      | 0       | −n*theta*P^n/(P^n + rhof) >,
>  <(−1 + rhof)*deltaM | −deltaM | 0 >,
>  <−n*rhob*(1+P^n)/B | 1/B     | −(P^n+n^2*rhob*P^n+rhof)/(B*(P^n + rhof)) >
> >;
```

$$J0 := \begin{bmatrix} -\theta\,(1+P^n) & 0 & -\dfrac{n\,\theta\,P^n}{\rho_f + P^n} \\ \delta_M\,(\rho_f - 1) & -\delta_M & 0 \\ \dfrac{n\,\rho_b\,(1+P^n)}{B} & \dfrac{1}{B} & -\dfrac{P^n + n^2\,\rho_b\,P^n + \rho_f}{B\,(P^n + \rho_f)} \end{bmatrix}$$

```
# The characteristic polynomial writes: a[0]*x^3 + a[1]*x^2 + a[2]*x + a[3]

> CP := normal(LinearAlgebra:−CharacteristicPolynomial(J0,x)):
```

```
> for i from 1 to LinearAlgebra:−RowDimension (J0) do
>     a[i] := coeff (CP, x, LinearAlgebra:−RowDimension (J0)−i)
> end do:

# The Hurwitz determinants

> c[0,0] := 1:
> c[1,0] := a[1]:
> c[2,0] := a[1]*a[2]−a[3]:
> c[3,0] := a[3]*(a[1]*a[2]−a[3]):

# If the three first determinants are positive then no Hopf bifurcation arises.

# The two first determinants are positive.
# The third one (c[2,0]) is a rational fraction with a positive denominator.
# Thus, if the numerator of c[2,0] is positive then no Hopf bifurcation arises.

> c[2,0] := normal (c[2,0]):
> c[2,0] := numer (c[2,0]);

                2        n 2
c[2, 0] := 2 n  rhob (P )  deltaM theta B rhof

          2       n                                    2             2
     + 2 n  rhob P  deltaM theta B rhof + deltaM rhof  + theta rhof

          2        n 2                               n 3    4     2   n 2
     + 2 n  rhob (P )  deltaM + 2 deltaM theta B (P )  + n  rhob  (P )  deltaM

        2        n 2                 n 3           n 2    2        n 3
     + n  rhob (P )  theta + theta (P )  + deltaM (P )  + n  rhob (P )  theta

            2       2         2       2           2    n
     + theta  B rhof  + deltaM  B rhof  + 2 deltaM  B P  rhof

            2  2   n 2               2     n 2             n
     + deltaM  B  (P )  theta + deltaM  B (P )  + 2 deltaM P  rhof

            2     n 4        2     n 2          2     n 3             n 2
     + theta  B (P )  + theta  B (P )  + 2 theta  B (P )  + 2 theta (P )  rhof

            n     2         2  2   n 3                2     n 2
     + theta P  rhof  + deltaM  B  (P )  theta + 4 theta  B (P )  rhof

            2  2   n 2                  2    n               2  2   n 3
     + theta  B  (P )  deltaM + 2 theta  B P  rhof + 2 theta  B  (P )  deltaM

                           2        2  2   n 4                  2       2  n
     + 2 deltaM theta B rhof  + theta  B  (P )  deltaM + 2 theta  B rhof  P

            2     n 3              2  2     2              2     n 2     2
     + 2 theta  B (P )  rhof + deltaM  B  rhof  theta + theta  B (P )  rhof

          2        n 3                          2         2       n
     + 2 n  rhob (P )  deltaM theta B + deltaM  B rhof n  rhob P

          2        n 2                                   n     2
     + 2 n  rhob (P )  deltaM theta B − deltaM theta B n P  rhof

                         n 2                            n
     − deltaM theta B n (P )  rhof + 4 deltaM theta B P  rhof

                        n             2  2     2
     + deltaM theta B n P  rhof + theta  B  rhof  deltaM

        2        n 2                                  n     2
     + n  rhob (P )  theta rhof + 2 deltaM theta B P  rhof
```

```
                          n 2           2       n
     + 4 deltaM theta B (P )  rhof + 2 n  rhob P  deltaM rhof

        2       n                       2  2   n 2
     + n  rhob P  theta rhof + 2 deltaM  B  (P )  theta rhof

             2     n 2  2                2  2  n
     + deltaM  B (P )  n  rhob + 2 deltaM  B  P  theta rhof

              2  2   n 3                      2  2     2         n
     + 2 theta  B  (P )  deltaM rhof + 2 theta  B  rhof  deltaM P

              2  2  n                      2  2   n 2
     + 2 theta  B  P  deltaM rhof + 4 theta  B  (P )  deltaM rhof

             2  2     2        n                      n 2
     + deltaM  B  rhof  theta P  + deltaM theta B n (P )

            2  2   n 2     2                              n 2           n 2
     + theta  B  (P )  rhof  deltaM + 2 deltaM theta B (P )  + theta (P )

               n
     + 2 theta P  rhof
```

```
# The negterms function takes a rational fraction expr as argument.
# It returns a list of two lists. The first (resp. second) list involves
# all the monomials occuring in the numerator (resp. denominator) of expr
# with a negative coefficient. See the AB 2007 paper, pages 66−80.

> negterms := proc (expr)
>     local f, p, koeffs, terms, result;
>     f := proc (x,y) if x < 0 then x*y else NULL fi end:
>     result := NULL;
>     for p in [expand (numer (expr)), expand (denom (expr))] do
>         if indets (p) <> {} then
>             koeffs := coeffs (p, indets (p), 'terms');
>             result := result, zip (f, [koeffs], [terms])
>         else
>             result := result, []
>         fi;
>     od;
>     [result]
> end:

# Since all variables and parameters are positive, if negterms (expr)
# is empty, then expr is positive. Very few monomials raise problems:

> outneg := negterms (c[2,0]):
> outneg := outneg[1]:
> factor(outneg[1]+outneg[2]);
                       n                          n
                    −P  deltaM theta B n rhof (P  + rhof)

# The numerator of c[2,0] is a polynomial in P^n.
# One performs the change of coordinates to get rid of n.

> cond := subs (P^n = P, c[2,0]):

# One has cond = d0*rhob^2 + d1*rhob + d2 with d0,d1 > 0.
# Thus d2 >= 0 if and only if c[2,0] > 0.

> d2 := coeff (cond, rhob, 0):

# A positive factor is removed.

> d2 := normal (d2 / (P + rhof));

            2    2               2  2  2                  2  2
d2 := theta  B P  rhof + 2 theta  B  P  deltaM + P deltaM  B  theta
```

```
                                    2  2
     + P deltaM theta B n + P theta  B  deltaM + 2 P deltaM theta B

               2                  2  2  3                  2             2
     + 2 theta  B P rhof + theta  B  P  deltaM + P theta  B + P deltaM  B

               2    2        2                2
     + 2 theta  B P  + theta  B rhof + deltaM  B rhof − deltaM theta B n P rhof

                                           2        2    3
     + P theta + P deltaM + 2 deltaM theta B P  + theta  B P  + theta P rhof

                          2  2                       2  2
     + theta rhof + deltaM  B  P theta rhof + 2 theta  B  P deltaM rhof

                          2  2  2                    2
     + deltaM rhof + theta  B  P  deltaM rhof + theta P

              2  2                   2  2                      2  2  2
     + deltaM  B  theta rhof + theta  B  deltaM rhof + deltaM  B  P  theta

     + 2 deltaM theta B P rhof + 2 deltaM theta B rhof

# One has d2 = e0*rhof + e1 with e1 > 0.
# Thus, e0 >=0 if and only if d2 >= 0 whence iff c[2,0] > 0

> e0 := coeff (d2, rhof, 1);

            2    2           2          2          2
e0 := theta  B P  + 2 P theta  B + theta  B + deltaM  B − P deltaM theta B n

                                  2  2                   2  2
     + P theta + theta + P deltaM  B  theta + 2 P theta  B  deltaM + deltaM

            2  2  2                2  2              2  2
     + theta  B  P  deltaM + deltaM  B  theta + theta  B  deltaM

     + 2 P deltaM theta B + 2 deltaM theta B

# One has e0 = f0*P^2 + f1*P + f2 with f0,f2 > 0.
# If e0 has no positive roots (in P) then e0 >= 0.
# Proof. Since f2/f0 = the product of the roots, the two roots
#        have the same sign and are nonzero.
#        For both roots to be positive, it is necessary and sufficient to have:
#        (*) f1 < 0 (since f1 = the opposite of the sum of the roots)
#        (*) discrim (cond, P) > 0 (in order to have real roots)

# Some positive factors are removed.

> f1 := normal (coeff (e0, P, 1)/theta);

                                          2  2           2
 f1 := 2 theta B − deltaM B n + 1 + deltaM  B  + 2 theta B  deltaM + 2 deltaM B

> disc := normal (discrim (e0, P)/theta^2);

               4  4           3  3             3       2             2  2
disc := deltaM  B  − 2 deltaM  B  n − 4 theta B  deltaM  n − 4 deltaM  B  n

               2  2  2           2  2           2
     + deltaM  B  n  − 2 deltaM  B  − 4 theta B  deltaM n − 2 deltaM B n + 1

# The two following conditions cannot be satisfied simultaneously if and
# only if c[2,0] is positive.
#   (*) 0 < theta < theta1
#   (*) 0 < theta < theta0
# where

> theta1 := solve (f1, theta);
```

```
                                  2  2
                         −deltaM  B  + deltaM B n − 1 − 2 deltaM B
                theta1 := ----------------------------------------
                                    2 B (1 + deltaM B)

> theta0 := solve (disc, theta);

                   4  4           3  3             2  2           2  2  2
theta0 := (deltaM  B  − 2 deltaM  B  n − 4 deltaM  B  n + deltaM  B  n

                  2  2                          /     2
     − 2 deltaM  B  − 2 deltaM B n + 1)  /  (4 B  deltaM n (1 + deltaM B))
                                       /

# Thus theta1 and theta0 cannot be positive simultaneaously if and only if
# c[2,0] is positive.

# One performs the following change of coordinates which gets rid of deltaM:

> theta1 := subs (deltaM=delta/B, theta1);

                                    2
                               −delta  + delta n − 1 − 2 delta
                     theta1 := -------------------------------
                                      2 B (1 + delta)

> theta0 := subs (deltaM=delta/B, theta0);

theta0 :=

          4          3            2          2  2          2
     delta  − 2 delta  n − 4 delta  n + delta  n  − 2 delta  − 2 delta n + 1
     ------------------------------------------------------------------------
                             4 B delta n (1 + delta)

# The denominators of theta0 and theta1 are positive. Thus the numerators
# cannot be positive simultaneaously if and only if c[2,0] is positive.
# One thereby gets rid of B.

> theta1 := numer (theta1);
                                    2
                     theta1 := −delta  + delta n − 1 − 2 delta

> theta0 := numer (theta0);

theta0 :=

          4          3            2          2  2          2
     delta  − 2 delta  n − 4 delta  n + delta  n  − 2 delta  − 2 delta n + 1

> theta0 := algsubs(delta^2 = lambda*delta−1, theta0);

                                  2       2
theta0 := (−2 n lambda + lambda ) delta

                                    2                                  2
     + (−4 lambda − 4 n lambda + n  lambda) delta + 4 n + 4 − n

> theta0 := algsubs(delta^2 = lambda*delta−1, theta0);

                                      2                3             2
theta0 := (−4 lambda − 4 n lambda + n  lambda + lambda  − 2 n lambda ) delta

               2                                  2
     − lambda  + 4 n + 2 n lambda + 4 − n

> theta0 := factor(theta0);

                          2                                  2
       theta0 := (lambda  − 4 n − 2 n lambda − 4 + n ) (lambda delta − 1)
```

```
> theta0 := normal (theta0/(lambda*delta−1));

                                  2                              2
                theta0 := lambda  − 4 n − 2 n lambda − 4 + n

> theta1 := algsubs(delta^2 = lambda*delta−1, theta1);

                       theta1 := (−lambda + n − 2) delta

> theta1 := normal(theta1/delta);

                            theta1 := −lambda + n − 2

# theta1 nonnegative implies n >= lambda + 2.
# theta0 and theta1 nonnegative imply n >= lambda + 2 + 2*sqrt(lambda+2).
# Since lambda = delta+1/delta, one has lambda >= 2 whence n >= 8.
# Thus c[2,0] is positive if and only if n < 8.

######################################################################
# LAST STEP : PROVING THE EXISTENCE OF BIFURCATIONS FOR n >= 9
######################################################################

# To have delta=1, one takes deltaM=1/2 and B=2 (B needs to be > 1).
# theta < theta0 and < theta1 leads to theta=1/20
# d2 < 0 leads to rhof=600
# the value of theta is obtained by solving cond=0

> n := 'n':

> valB := 2:
> valP := 10:
> valdeltaM := 1/2:
> valtheta := 1/20:
> valrhof := 600:

> values := B=valB, deltaM=valdeltaM, theta=valtheta, rhof=valrhof, P=valP:

> valrhob := normal (solve (subs (values,cond), rhob) [1]);

                                                        1/2
                              −3843 + (1640961 + 1461560 n)
                  valrhob := −−−−−−−−−−−−−−−−−−−−−−−−−−−−−−
                                          2
                                       20 n

# The following commands show that c[0,0] > 0 and c[1,0] > 0 and c[2,0] = 0
# with c[2,1] = −a[3] < 0 : a Hopf bifurcation

> values := values [1..−2], rhob=valrhob, P=valP^(1/n):

# Let us check that c[2,0] is always zero
> simplify (subs (values, c[2,0]), assume=positive);

                                        0

# Let us check that −a[3] is always negative
> simplify (subs (values, −a[3]), assume=positive);

                                     11   599 n
                                  − −− − −−−−−
                                     80   4880

> J1 := simplify (subs (values, J0), assume=positive):

# Let us see the Hopf on the eigenvalues of the linearized differential system:
# one eigenvalue is real and negative, the two other eigenvalues are complex
# and conjugated with a zero real part.

> n := 15:
```

```
> evalf (LinearAlgebra:−Eigenvalues (J1));

                                [1.003611679 I ]
                                [               ]
                                [−1.003611679 I]
                                [               ]
                                [ −1.964472807 ]

# Let us now see the Hopf on the nonlinear differential system.
# For this, one needs to compute the values of some variables and parameters.

# To be recomputed since the value of n may be changed.

> denominator :=
>     subs (K_0 = 1,
>       add ((i+1)^2*cat(K_,i)*P(t)^i*alpha^i, i=0..n−2)+
>         n^2*P(t)^(n−1)*alpha^(n−1)):

> PaperReducedModel :=
>     [ Diff (G(t),t) = rhob*(gamma0 − G(t) − G(t)*P(t)^n),
>       Diff (M(t),t) = (rhof*G(t) − G(t) + gamma0 − M(t))*deltaM,
>       Diff (P(t),t) =
>       (n*theta*(gamma0 − G(t) − G(t)*P(t)^n) + M(t) − P(t))/denominator
>     ]:

> SteadyPoint :=
>     gamma0 = P*(1 + P^n)/(P^n + rhof),
>          M = P,
>          G = P/(P^n + rhof):

# Let us check that the paper's formula above is correct.
# The following command should print a list of zeros.

> simplify (subs(SteadyPoint, MABSys:−SteadyPointSystem(PaperReducedModel)));

                                  [0, 0, 0]

# Let us fix the values of the K_i's to 1
> values := seq (cat(K_,i)=1, i=1..n), values:

# Let us compute the value of alpha such that the denominator
# is equal to valB on the steady point
> valalpha :=
>     fsolve (subs (P(t)=P, values, denominator)=valB) [−1];

                           valalpha := 0.1414148855

# gamma0 is provided by the steady point equations.

> valgamma0 := subs (SteadyPoint, values, gamma0);

                                                1/15
                                           11 10
                            valgamma0 := ---------
                                            610

> valG := subs (SteadyPoint, values, G);

                                             1/15
                                          10
                                valG := ------
                                           610

> valM := subs (SteadyPoint, values, M);

                                           (1/15)
                                valM := 10

> values := alpha=valalpha, gamma0=valgamma0, values [1..−2]:
```

```
# Let us check that values is correct. The eigenvalues of J2 should be
# the same as that of J1 computed formerly.

> J2 := simplify (subs (values, J0), assume=positive):
> J2 := subs( P=valP^(1/n), J2):
> evalf (LinearAlgebra:−Eigenvalues (J2));

                                [1.003611679 I ]
                                [              ]
                                [−1.003611679 I]
                                [              ]
                                [ −1.964472807 ]


# In order to cross the bifurcation point and see some oscillating
# behaviours, one perturbates the values by dividing rhob by 2

> values := deltaM = valdeltaM,
>         theta = valtheta,
>         rhof = valrhof,
>         rhob = valrhob/2,
>         gamma0 = valgamma0,
>         alpha = valalpha,
>         seq (cat(K_,i)=1, i=1..n);

                                                                    1/2
                                                  427    23564361
values := deltaM = 1/2, theta = 1/20, rhof = 600, rhob = − ---- + ----------,
                                                  1000      9000

                  1/15
              11 10
    gamma0 = ---------, alpha = 0.1414148855, K_1 = 1, K_2 = 1, K_3 = 1,
                 610

    K_4 = 1, K_5 = 1, K_6 = 1, K_7 = 1, K_8 = 1, K_9 = 1, K_10 = 1, K_11 = 1,

    K_12 = 1, K_13 = 1, K_14 = 1, K_15 = 1

# Let us check that the values are all positive
> evalf(values);

deltaM = 0.5000000000, theta = 0.05000000000, rhof = 600., rhob = 0.1123681832,

    gamma0 = 0.02102468593, alpha = 0.1414148855, K_1 = 1., K_2 = 1., K_3 = 1.,

    K_4 = 1., K_5 = 1., K_6 = 1., K_7 = 1., K_8 = 1., K_9 = 1., K_10 = 1.,

    K_11 = 1., K_12 = 1., K_13 = 1., K_14 = 1., K_15 = 1.

# Let us check that the steady point has become unstable:
# the real parts of the two complex eigenvalues should be positive.

> J3 := simplify (subs (values, J0), assume=positive):
> J3 := subs(P=valP^(1/n), B=valB, J3):
> J3 := evalf(J3):
> evalf (LinearAlgebra:−Eigenvalues (J3));

                        [0.04525661289 + 1.033835489 I]
                        [                              ]
                        [        −1.847749628          ]
                        [                              ]
                        [0.04525661289 − 1.033835489 I]

# Let us integrate the solution

# The initial values of the dep. variables are taken "near" the steady
# point of the non perturbated system
```

```
> syst := subs (values, PaperReducedModel);

                    /                  1/2\ /     1/15                     \
          d         |  427    23564361   | |11 10                      15|
syst := [-- G(t) = |- ---- + -----------| |--------- - G(t) - G(t) P(t)  |,
          dt        \  1000      9000    / \   610                       /

                                1/15                        /
    d                         11 10                     d   |
    -- M(t) = 599/2 G(t) + --------- - 1/2 M(t), -- P(t) = |
    dt                          1220                    dt  \

         1/15                                          \
    33 10                                  15          |   /
    --------- - 3/4 G(t) - 3/4 G(t) P(t)   + M(t) - P(t)|  /  (1
      2440                                             / /

                                          2                     3
     + 0.5656595420 P(t) + 0.1799835286 P(t)  + 0.04524862237 P(t)

                           4                      5                      6
     + 0.009998169925 P(t)  + 0.002036001679 P(t)  + 0.0003918923965 P(t)

                          7                       8
     + 0.00007238454643 P(t)  + 0.00001295525688 P(t)

                       -5     9                 -6     10
     + 0.2261810084 10   P(t)  + 0.3870228731 10   P(t)

                       -7     11                -7     12
     + 0.6513416958 10   P(t)   + 0.1081006286 10   P(t)

                       -8     13                -9     14
     + 0.1772934586 10   P(t)   + 0.2878155706 10   P(t)  )]

> sol := dsolve
>       ({op(syst), P(0)=valP^(1/n)+1e-7, M(0)=valM+1e-6, G(0)=valG-1e-7},
>        {G(t),M(t),P(t)}, numeric);

                       sol := proc(x_rkf45)  ...  end proc

# The following commands plot curves which actually show oscillations.
# An example is provided in the paper.

> plotsetup (x11);
> plots[odeplot] (sol, [t,G(t)], t = 0..400, numpoints=100000);
> plots[odeplot] (sol, [t,M(t)], t = 0..400, numpoints=100000);
> plots[odeplot] (sol, [P(t),G(t)], t = 170..300, numpoints=100000);
> plots[odeplot] (sol, [M(t),G(t)], t = 170..300, numpoints=100000);
> plots[odeplot] (sol, [M(t),P(t)], t = 170..300, numpoints=100000);

# Paper plotting

> values := deltaM=1/2, theta=1/20, rhof=600, rhob=1/40,
> gamma0=1/50, alpha=6/25, seq(cat(K,'_',i)=1,i=1..n);

values := deltaM = 1/2, theta = 1/20, rhof = 600, rhob = 1/40, gamma0 = 1/50,

    alpha = 6/25, K_1 = 1, K_2 = 1, K_3 = 1, K_4 = 1, K_5 = 1, K_6 = 1, K_7 = 1,

    K_8 = 1, K_9 = 1, K_10 = 1, K_11 = 1, K_12 = 1, K_13 = 1, K_14 = 1,

    K_15 = 1

> sol := dsolve
>       ({op(syst), P(0)=1.17, M(0)=0.8, G(0)=0.002},
>        {G(t),M(t),P(t)}, numeric);

                       sol := proc(x_rkf45)  ...  end proc
```

```
> plotsetup (x11);
> plots[odeplot] (sol, [t,G(t)], t = 0..400, numpoints=100000);
> plots[odeplot] (sol, [t,M(t)], t = 0..400, numpoints=100000);
> plots[odeplot] (sol, [P(t),G(t)], t = 0..300, numpoints=100000);
```