|  |  |
| --- | --- |
| Note: | PGO-2021-05 v3 |
| From: | Paul W. Goedhart (Biometris, Wageningen UR) |
| To: | Jan Huijsmans (WPR), Dennis te Beest (Biometris) and Sasha Hafner (Aarhus University) |
| Topic: | The ALFAM2 dynamic mathematical model applied to Dutch data |
| Date: | 16-02-2024 |
| #Pages: |  |

# Introduction

Hafner *et al* (2019) describe a semi-empirical dynamic mathematical model, further called ALFAM2, for estimating ammonia volatilization from field applied manure. In ALFAM2 applied TAN is immediately partitioned between two pools: a “fast” pool and a “slow” pool. Emission of TAN from these pools, and transfer from the fast to the slow pool, is described by a first-order process with different rates. The model thus describes emission in continuous time. The partition parameter and the rates may depend on temperature and wind speed, on the grass height or the soil of the field, or on manure characteristics such as pH or dry matter content. The model was fitted to the ALFAM2 database which contains measurements of NH­3 emission from field-applied manure in 13 countries. Since these are time-interval measurements, especially for temperature and wind speed, this implies that a discretized version of the continuous model was fitted to the data. Application of the fitted model to Dutch field experiments revealed that the ALFAM2 model was unable to reproduce observed emission percentages.

In this note the ALFAM2 model is used and extended in the following ways:

1. A sink pool is added to the model to ensure that possibly, and very likely, not all TAN will be emitted. The sink pool is filled by a first-order process from the slow pool.
2. A typical time-interval sequence in the Dutch field experiments is 1-2-3-6-12-24-24-24 hours giving 96 hours in total. The ALFAM2 database contains temperature and wind speed data for these intervals. However the mean temperature over a prolonged time interval of say 24 hours may not adequately reflect the dependency of NH3 emission on temperature. Therefore, here hourly temperature and wind speed data are used to fit the model.
3. The model is fitted to a large set of Dutch field experiments, separately for the three application method broadcast surface spreading (BS), narrow band application (NB) and shallow injection (SI).

# The ALFAM2 model

Slow Pool (S)

Sink Pool (Z)

Applied

TAN

Fast Pool (F)

Emission (G)

Emission (T)

Figure 1: *Structure of the ALFAM2 model.*

The ALFAM2 model, graphically depicted in Figure 1, describes the amount of TAN over time in the fast pool , the slow pool , the sink pool , the emission from the fast pool and the emission from the slow pool . Initially, at =0, total applied TAN is immediately partitioned between the fast and the slow pool giving initial conditions

, , , , (1)

The rate of change in each of the five pools is described by the five differential equations

(2)

It is convenient to define the following symbols , and . The solution of the differential equations is then given by

(3)

Note that the solution for is identical to equation (3) in Hafner *et al* (2019) and the solution for is equivalent, i.e. now incorporating , to their equation (7). The solution for follows from the solution for and obeys the initial condition . The solution for and follow from the solution for and respectively.

The expression for in equation (3) involves division by which can equal zero. However the limit of for equals . The solution for , and then becomes

(4)

The amount of TAN emitted at time is given by the amount not in either of the three compartments, i.e. , and this equals . A proof of this equality is given in Appendix A. An implementation in R is given in Appendix B.

For time-interval measurements, the four rate parameters (, , , ) may depend on time-interval temperature and wind speed measurement. In that case, in each time-interval of length , a discretized version of the solutions in equation (3) can be used where implicitly the rate parameters can be different for each time-interval

(5)

The sequence , , … , starts with the initial values given in equation (1). Note that , i.e. the amount in pool at the cumulative time , and similarly for , , and . This should be obvious; in case it is not Appendix C gives a simple numeric example.

Linking the primary parameters to predictor variables should take account of the fact that the primary rate parameters are non-negative and the initial partition parameter is in the interval (0,1). This is accomplished by employing an appropriate link function. With predictor variables , …, and linear predictors we employ and . In the implementation is bounded by limiting to the interval , and is bounded by constraining to the interval . These limits can be modified by a “control” list.

In case there is only a single emission pool ­ along with a sink pool , a possible simple model is given in Figure 2. In this model, a proportion of TAN is immediately transferred to the sink and will not be emitted. So the starting conditions are and The solution of the differential equations is given by

(6)

Sink Pool (Z)

Applied

TAN

Pool (P)

Emission (E)

Figure 2: *Structure of the ALFAM2 model with a single pool and a sink pool .*

The derivative of in equation (6) equals which is in accordance with the structure of the model in Figure 2.

1. Equality of expression for emission at timepoint

Emission at time equals where

(3)

Terms on the left and right side of the equality which are a multiple of :

|  |  |
| --- | --- |
|  | Re-arranging and noting that gives |
|  | Combining terms in brackets gives |
|  | This is true since and terms cancel |

Terms on the left and right side of the equality which are a multiple of :

|  |  |
| --- | --- |
|  | Division by the common term gives |
|  | This is true since |

Terms on the left and right side of the equality which do not involve or :

|  |  |  |
| --- | --- | --- |
|  | | |
|  | So terms with cancel out, division by gives | |
|  | Using a common denominator gives | |
|  | | Combining terms gives |
|  | | Simplification gives |
|  | | And this is true |

This completes the proof

1. R function for the ALFAM emission model

***##### Function for ALFAM emission model***

***# INPUT***

***#   t    : vector of time points***

***#   par  : list of scalar parameters r1, r2, r3, r4, r12, r34, ra***

***#   init : list of scalar initial values F0, S0, Z0, G0, T0***

***#   eps\_ra : scalar for branching on ra values close to zero***

***# OUTPUT dataframe with***

***#   t    : vector of time points***

***#   Ft   : amount in the fast pool***

***#   St   : amount in the slow pool***

***#   Zt   : amount in the sink pool***

***#   Gt   : amount emitted from the fast pool including G0***

***#   Tt   : amount emitted from the slow pool including T0***

***#   Et   : total amount emitted including G0 and T0 (equals Gt+Zt)***

***#   Eti  : total amount emitted excluding G0 and T0***

***# REMARK***

***#   When the function is called for successive time intervals***

***#   Eti is the amount emitted in the time interval***

**Fitted** **<-** **function**(t, par, init, eps\_ra**=**1.0e-6) {

***##### Prepare; par and init should contain scalars***

**if** (length(unlist(par)) **!=** 7) stop("Argument par should contain 7 scalars.")

**if** (length(unlist(init)) **!=** 5) stop("Argument init should contain 5 scalars.")

  exp12 **<-** exp(**-**par**$**r12**\***t)

  exp34 **<-** exp(**-**par**$**r34**\***t)

  F0r2  **<-** init**$**F0 **\*** par**$**r2

***##### Calculate functions***

  Ft **<-** init**$**F0 **\*** exp12

  Gt **<-** init**$**G0 **+** init**$**F0**\***(par**$**r1**/**par**$**r12)**\***(1 **-** exp12)

**if** (abs(par**$**ra) **>=** eps\_ra) {

    tmp1  **<-** F0r2**/**par**$**ra

    tmp2  **<-** init**$**S0 **+** tmp1

    St  **<-** tmp2**\***exp34 **-** tmp1**\***exp12

    tmp **<-** tmp2**\***(1 **-** exp34)**/**par**$**r34 **-** tmp1**\***(1 **-** exp12)**/**par**$**r12

  } **else** {

    St  **<-** (init**$**S0 **+** F0r2**\***t) **\*** exp34

    tmp **<-** **-**(init**$**S0**\***(exp34**-**1) **+** F0r2**\***(exp34**\***(par**$**r34**\***t**+**1)**-**1)**/**par**$**r34)**/**par**$**r34

  }

  Zt **<-** init**$**Z0 **+** par**$**r4**\***tmp

  Tt **<-** init**$**T0 **+** par**$**r3**\***tmp

  Et **<-** Gt **+** Tt

**return**(data.frame(t, Ft, St, Zt, Gt, Tt, Et, Eti**=**Et **-** init**$**G0 **-** init**$**T0))

}

1. Solution for time-interval measurements is equivalent

***##### Call the function for a time vector***

init **<-** list(F0**=**75, S0**=**25, Z0**=**0, G0**=**0, T0**=**0)

par  **<-** list(r1**=**0.05, r2**=**0.1, r3**=**0.03, r4**=**0.15)

par**$**r12 **<-** par**$**r1**+**par**$**r2;  par**$**r34 **<-** par**$**r3**+**par**$**r4;  par**$**ra **<-** par**$**r12**-**par**$**r34

time **<-** c(0, 1, 3, 6, 10, 12, 15, 18, 20, 40, 80)

fit1 **<-** Fitted(time, par, init)

print(round(fit1, 2), row.names**=**F)

    t      Ft      St      Zt      Gt      Tt      Et     Eti

    0   75.00   25.00    0.00    0.00    0.00    0.00    0.00

    1   64.55   27.24    3.94    3.48    0.79    4.27    4.27

    3   47.82   28.29   12.36    9.06    2.47   11.53   11.53

    6   30.49   25.23   24.53   14.84    4.91   19.74   19.74

   10   16.73   18.59   37.71   19.42    7.54   26.96   26.96

   20    3.73    6.30   55.18   23.76   11.04   34.79   34.79

***##### Call the function for successive time intervals while updating init***

interval **<-** diff(time)

print(interval)

1  2  3  4 10

fit2 **<-** NA **\*** fit1

fit2[1,] **<-** c(0, init, 0, 0)

row **<-** 1

**for** (tt **in** interval) {

  fit **<-** Fitted(tt, par, init)

  row **<-** row **+** 1

  fit2[row,] **<-** fit

  init**$**F0 **<-** fit**$**Ft

  init**$**S0 **<-** fit**$**St

  init**$**Z0 **<-** fit**$**Zt

  init**$**G0 **<-** fit**$**Gt

  init**$**T0 **<-** fit**$**Tt

}

print(round(fit2, 2), row.names**=**F)

    t      Ft      St      Zt      Gt      Tt      Et    Eti

    0   75.00   25.00    0.00    0.00    0.00    0.00   0.00

    1   64.55   27.24    3.94    3.48    0.79    4.27   4.27

    2   47.82   28.29   12.36    9.06    2.47   11.53   7.26

    3   30.49   25.23   24.53   14.84    4.91   19.74   8.21

    4   16.73   18.59   37.71   19.42    7.54   26.96   7.22

   10    3.73    6.30   55.18   23.76   11.04   34.79   7.83

maxDiff **<-** max(abs(unlist(fit1[,2**:**6] **-** unlist(fit2[,2**:**6]))))

print(maxDiff)

2.842171e-14