ETH
Eidgenössische Technische Hochschule Zürich
Swiss Federal Institute of Technology Zurich

Lecture with Computer Exercises:
Modelling and Simulating Social Systems with MATLAB

Project Report

Hive Simulation

Stefan Gugler & Elias Huwyler & Fabian Tschopp

Zurich
December 2013

Agreement for free-download

We hereby agree to make our source code for this project freely available for download from the web pages of the SOMS chair. Furthermore, we assure that all source code is written by ourselves and is not violating any copyright restrictions.

Stefan Gugler Elias Huwyler Fabian Tschopp

ETH

Eidgenössische Technische Hochschule Zürich
Swiss Federal Institute of Technology Zurich

Declaration of Originality

This sheet must be signed and enclosed with every piece of written work submitted at ETH.

I hereby declare that the written work I have submitted entitled

Hive Simulation

is original work which I alone have authored and which is written in my own words.*

Author(s)

Last name	First name
Gugler, Huwyler, Tschopp	Stefan, Elias, Fabian

Supervising lecturer

Last name	First name
Kuhn, Woolley	Tobias, Olivia

With the signature I declare that I have been informed regarding normal academic citation rules and that I have read and understood the information on 'Citation etiquette' (http://www.ethz.ch/students/exams/plagiarism_s_en.pdf). The citation conventions usual to the discipline in question here have been respected.

The above written work may be tested electronically for plagiarism.

ETH Zurich, 12.12.2013
Place and date

Signature

*Co-authored work: The signatures of all authors are required. Each signature attests to the originality of the entire piece of written work in its final form.

Print form

Abstract

In this article, we present a model that simulates the behaviour of a single hive of honey bees (*Apis mellifera*), interacting with its environment. Three different kinds of bees were implemented: The forager, the scout bee and the hive bee. The hive is a quantitative model substantially based on the works of *David S. Khoury et al.* [1] using a range of differential equations to represent hive growth, social inhibition, food availability and bee ageing. The focus of this model lies, in contrast to *Khoury et al.*, in the hive's interaction with the environment and it's dependency on the seasonal changes (no food income in winter, different flowers each month). To simulate seasons, a fine-grained and easily extensible environment algorithm is used, in which the bees navigate and collect food. For the development of the environment simulation, empirical observations by *T.D. Seeley* [2] were used.

Contents

1 Individual contributions

We discussed and extended the model together on a weekly basis, collecting and mixing individual ideas. The report was written in a cooperative manner. Every team member wrote the sections according to their *individual tasks* and then corrected and extended the sections of the other two members. The individual tasks have been assigned by taking into account the programming skills, personal interests and available time.

Individual tasks:

- *Stefan Gugler, BSc Interdisciplinary Sciences:* Interpretation and application of empirical data in the basic and advanced model. Searching references and data in the literature.
- *Elias Huwyler, BSc Computer Science:* MATLAB scripts for map generation, map loading and data plotting. Interpretation and analysis of the simulation results.
- *Fabian Tschopp, BSc Computer Science:* Implementation of the basic and advanced model in MATLAB (object oriented structure, code hierarchy, actual implementation).

2 Introduction and Motivations

The global decrease of honey bee populations (*Apis mellifera*) is not only conspicuous, but also alarming. Our dependency on pollination in almost all forms of agricultural cultivation (such as fruit and vegetables, spices, nuts etc.) urges us to find out more about the bee's behaviour and their mortality. In recent years, many studies [3] [4] have been carried out to find out the reasons for the mass decline of pollinators: diseases, destruction of the environment, pesticides/fungicides (such as imidacloprid, deltamethrin [5], fipronil [6], and many others) and maybe even climate change [7]. The sum of all these factors culminates in the mass death of bees and can lead to a serious economic crash as well as to a impairment of agricultural resources. This motivated us to develop a simulation to find corner situations in which the hive's health becomes unstable and does not survive. We started with an existing simulation by *Khoury et al.* [1] and extended it with knowledge and empirical data from *T.D. Seeley* [2].

3 Description of the Model

3.1 Basic model: Hive simulation

Our model is widely based on the studies and equations of *Khoury et al.* [1]. We used his quantitative models (equation 1 to 6) as described bellow and added new parameters to find out more about the bee's behaviour.
Food, seasons, brood, foragers and hive bees are the cornerstone of our model. The dynamics of the hive is based on the bee's behaviour and their interaction with the environment as well as natural influences, i.e. seasons (which are separately added to *Khoury's* model). Food here means nectar and pollen, which doesn't need to be further distinguished after according to *Khoury* for simplicity reasons. After the queen laid an egg, a larvae develops inside a honeycomb cell. The equations show the proportionality of the brood to the food income and the relations between the bee number. Neglecting the complex process of reaching adulthood, we assume, according to *Khoury* that larvae become adult hive bees 12 days after pupation. The mortality rate of hive bees or capped brood is negligible if there are not specific diseases implemented. The agents are only female bees, since they are responsible for maintenance and sustainability of the hive and foraging process. The males are ordinary hive bees. With all these information we can set up the equation for the brood number change:

$$\frac{dB}{dt} = LS(H, f) - \varphi B \tag{1}$$

It associates different factors to the brood number. L is rate of the bee queen and $S(H, f)$ is a function of the survival rate dependent on the amount of food and the number of hive bees. φ is the adult bee emerging factor and B represents the uncapped brood. The equation gives us the survival rate of the brood B. $S(H, f)$ with H representing the number of hive bees and f the amount of food, is modelled as following:

$$S(H, f) = \frac{f^2}{f^2 + b^2} \frac{H}{H + v} \tag{2}$$

The variable v indicates the effect of the hive bees on the brood, whereas b shows the food effect on brood survival. It decreases as food increases. As f and H become very large, $S(H, f)$ becomes constant. The first factor is a sigmoid function and shows the correlation of the food available and the capped brood. A decrease in brood can occur because of lack of food (they cannot be fed) and / or because adult bees consume the larvae to keep the resources in the hive and recycle the proteins (cannibalism). The second factor models the interdependency of the hive

bee numbers on the survival of the brood. If there are large stores of food but no hive bees that can provide the food to the larvae, the brood survival also declines.
The second differential equation describes the rate at which the number of hive bees changes.

$$\frac{dH}{dt} = \varphi B(t-\tau) - HR(H,F,f) \tag{3}$$

τ is the ageing time and $\varphi B(t-\tau)$ is the rate at which adult bees develop from pupation to adult bees (emergence rate). H is again the hive bee number, the last term is the recruitment function and given as

$$R(H,F,f) = \alpha_{min} + \alpha_{max}(\frac{b^2}{b^2+f^2}) - \sigma(\frac{F}{F+H}) \tag{4}$$

This function models the change from a hive bee to a forager bee. α_{min} denotes the transition when there is enough food but not much foragers. Vice versa, α_{max} is used for less food. As before, b is the food effect on brood survival and f is the amount of food. σ is the strength of social inhibition and is dependent on the number of foragers and hive bees.
The third differential equation describes correspondingly the rate at which the number of foragers changes:

$$\frac{dF}{dt} = HR(H,F,f) - mF \tag{5}$$

The mortality rate mF indicated the rate at which the foragers F die. The recruitment function is the same as above.
The last equation outlines with the rate of food change in the hive's stores. Forager bees and hive bees are treated equally because their consumption is almost the same.

$$\frac{df}{dt} = cF - \gamma_A(F+H) - \gamma_B B \tag{6}$$

where c describes the average food a single forager collects per day.γ is the consumption of adult bees (A) and brood (B). In our extended model, we implemented an actual environment simulation so that the food income changes over time (see chapter 3.3). Equation 6 is then replaced by equation 11, which takes the daily food income instead of a fixed rate c per forager.

With those four differential equations, the change of brood (1), hive bee number (3), foragers (5) and change of food store (6) and the functions of the brood survival rate, correlating to the hive bee number and the food store (2) and a recruitment function (4), we can simulate a basic model of a hive with its interactions and changes. So far, it is identical with *Khoury's* model.

3.2 Changing laying rate over seasons

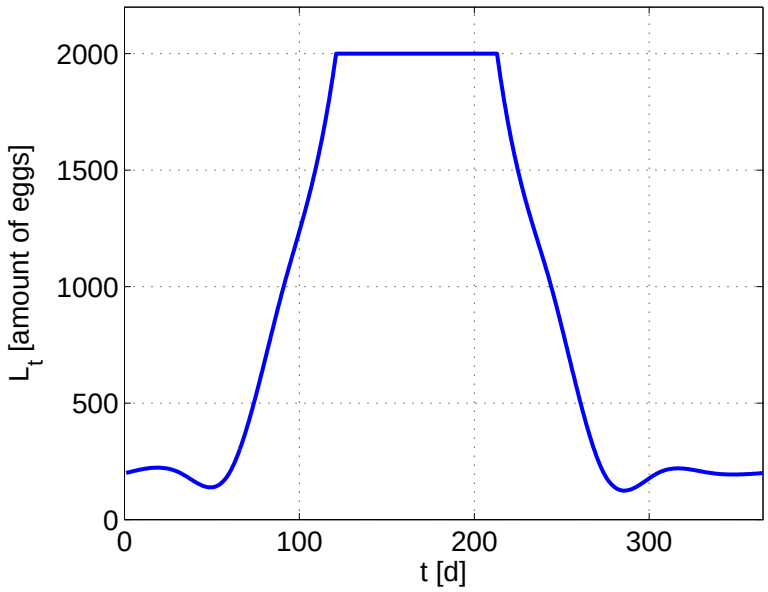


Figure 1: *The laying rate of the bee queen plotted over a year. The amount of eggs is capped after 2000 eggs.*

To make the model more realistic, equation 1 is replaced by equation 7. This incorporates that the bee queen does not have a constant laying rate. Instead, interpolated values based on *Wisdom of the Hive*, p. 34 [2] are used for the laying rate L_t (see fig. 1). *T.D. Seeley* denotes that the queen starts reproducing by early spring. We assumed that reproduction between April/May and October is realistic, as we obtained similar numbers of bees in summer (maxima) and winter (minima) to the empirical data of *T.D. Seeley* [2]. The minimal laying rate of 200 eggs per day is used to compensate mortality during winter. The peak of the laying rate (2000 eggs per day) is the same as used by *Khoury et al.* [1] and *Wright* [8]. The results are discussed in chapters 5.1 and 5.2 with the corresponding plots in appendix A.1.2 and A.1.3.

$$\frac{dB}{dt} = L_t S(H, f) - \varphi B \tag{7}$$

The introduction of dynamic laying rates led to problems with the mortality rates tested by *Khoury et al.* [1]. We discuss this in chapter 5.2.

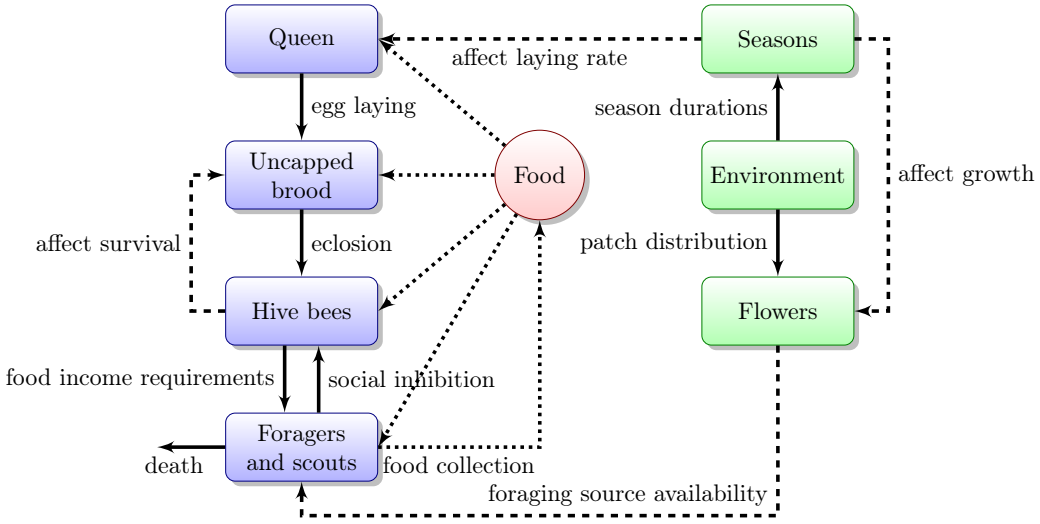


Figure 2: *Advanced honey bee social dynamics covered by this model.*

3.3 Advanced model: Environment simulation

As we were interested in observing the social dynamics under different food availability conditions, in contrast to the model of *Khoury et al.* [1], we had to implement a simulation for the environment. This advanced model introduces an environment with seasons which affect the queen's laying rate (see equation 7), the distribution and quality of flower patches and seasonal flower growth. The forager bees are then simulated on the environment, searching and collecting food according to the rules in chapter 3.3.1. An overview of the complete model is given in Figure 2.
Note that the environment simulation is triggered for day t after the basic model has computed all differential equations for day $t-1$. If there is no flower blooming on day t, we assume the daily food income $f_{d,t} = 0$ (see equation 11) and skip the environment simulation for this day.

3.3.1 Bees' working states

The environment simulation only simulates forager and scout bees. Hive bees are not considered as agents. For the foragers and scouts, we developed a set of simple rules which try to be as close as possible to the foraging process described by *T.D. Seeley* [2]. All essential rules are represented in Figure 3.

At the beginning of every day ($t_s = 0$), all bees are unemployed. There are three possibilities for every unemployed bee which are considered in every simulation step:

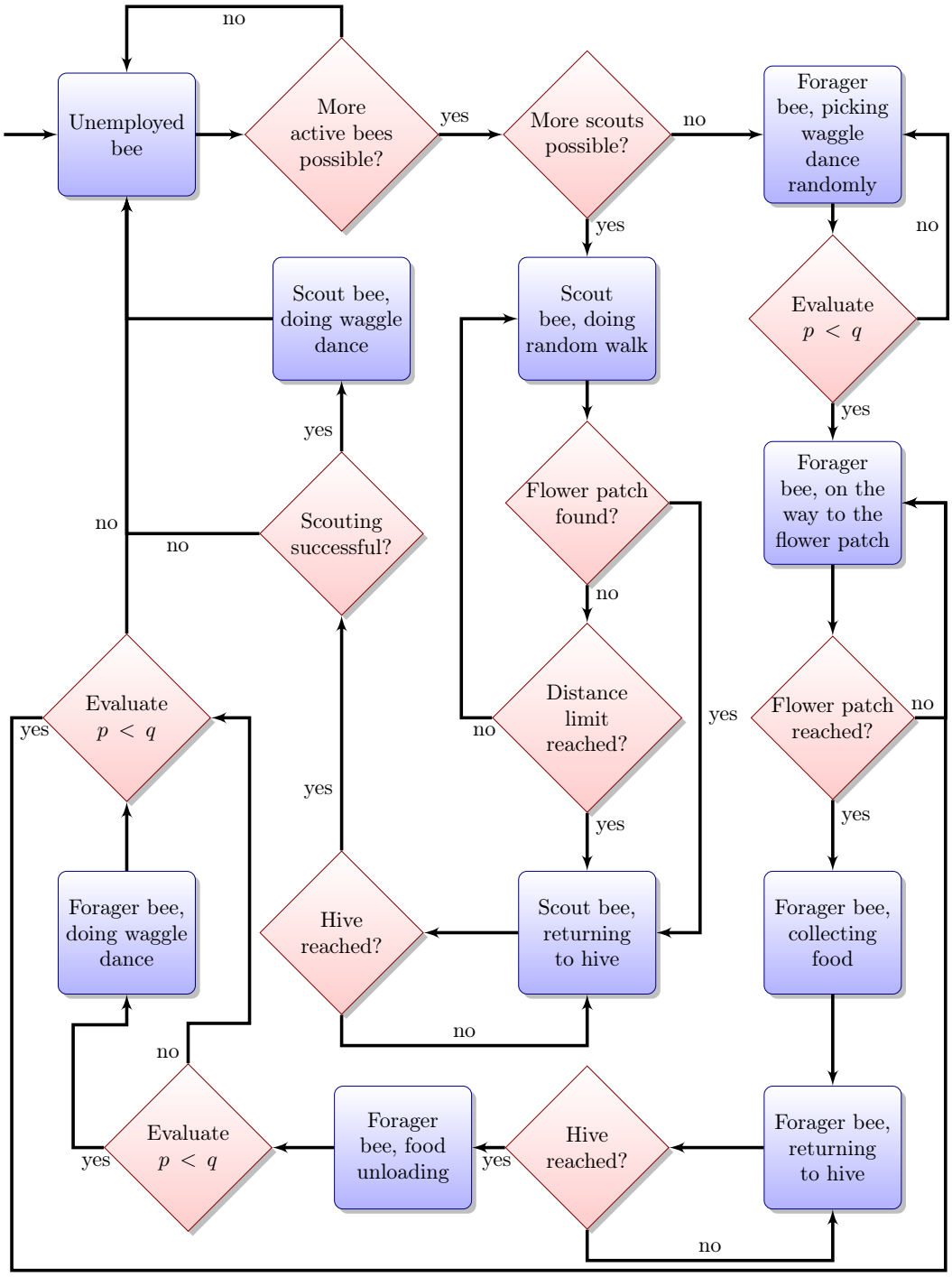


Figure 3: *State transitions of the bees in the environmental simulation.*

- If the amount of active bees according to the current time of the day is not reached, the bee can get a job assigned. Otherwise the bee stays unemployed.
- If the maxima of scouts is not reached (6% of the bees' foragers [2], p. 86), then the bee becomes a scout.
- Otherwise, the bee becomes a forager.

When a bee agent gets a scouting job assigned, the agent continues the following way:

- The scout performs a random walk (see chapter 4.3.1) and checks with every iteration step, if any flower patch has been passed (see chapter 4.3.2).
- If a flower patch is found or the flight distance limit is reached, the bee returns to the hive. Path optimization (see chapter 4.4.1) may occur.
- Back in the hive, the bee performs a waggle dance if scouting has been successful and goes back to the unemployed state.

When a bee agent gets a foraging job assigned, the agent continues the following way:

- Among the available waggle dances, the forager chooses randomly and evaluates if the food source is good enough ($p < q$) to visit or not (see chapter 3.3.2). This is repeated in every iteration step until the forager decides to visit the flower patch represented by the chosen waggle dance.
- The forager flies to the patch and, once reached, collects as much food as the flowers in the patch allow (see chapter 3.3.3).
- After foraging, the bee returns to the hive. Path optimization may occur again.
- Back in the hive, the forager unloads the collected food (increasing $f_{d,t}$, equations 10 and 11).
- The bee now has to evaluate the food source several times (($p < q$), once per iteration step) and decide if it is worth reporting the patch (executing a waggle dance) or not.
- A final evaluation ($p < q$) has to be done to decide if the forager will abandon it's own food source and become unemployed or fly to the same patch again.

3.3.2 Foragers' distribution across flower patches

It is clear from chapter 3.3.1 that after some time, the bees will focus on newly reported and then on the most profitable food sources because of relative quality evaluation. $p \in [0,1]$ denotes an equally distributed random value. q denotes the relative quality of a given flower patch compared to all patches currently available as a waggle dance. In detail, the evaluation ($p < q$) works as follows:

At the beginning of every iteration step, the waggle count n_w is set to zero. Every bee that does a waggle dance at the moment then writes the flower quality b_w, distance to the flower patch d_w and size of the patch A_w into the corresponding vector. This gives three vectors:

$$\begin{pmatrix} b_{w,1} \\ b_{w,2} \\ \dots \\ b_{w,n_w} \end{pmatrix} \qquad \begin{pmatrix} d_{w,1} \\ d_{w,2} \\ \dots \\ d_{w,n_w} \end{pmatrix} \qquad \begin{pmatrix} A_{w,1} \\ A_{w,2} \\ \dots \\ A_{w,n_w} \end{pmatrix}$$

To compare a (potential) waggle dance with given b_w, d_w and A_w against the existing ones, the following formula is used:

$$q_w = b_w \cdot \max(\begin{pmatrix} b_{w,1} \\ b_{w,2} \\ \dots \\ b_{w,n_w} \end{pmatrix})^{-1} \cdot d_w \cdot \max(\begin{pmatrix} d_{w,1} \\ d_{w,2} \\ \dots \\ d_{w,n_w} \end{pmatrix})^{-1} \cdot A_w \cdot \max(\begin{pmatrix} A_{w,1} \\ A_{w,2} \\ \dots \\ A_{w,n_w} \end{pmatrix})^{-1} \tag{8}$$

We update and obtain a fourth vector, every time a new waggle dance is executed, keeping track of all relative patch qualities. Every time a bee decides to do a waggle dance in the current iteration step, the update is computed:

$$\begin{pmatrix} q_{w,1} \\ q_{w,2} \\ \dots \\ q_{w,n_w} \\ q_{w,n_w+1} \end{pmatrix} = \begin{pmatrix} q_{w,1} \\ q_{w,2} \\ \dots \\ q_{w,n_w} \\ q_w \end{pmatrix} \cdot \max(\begin{pmatrix} q_{w,1} \\ q_{w,2} \\ \dots \\ q_{w,n_w} \end{pmatrix}, q_w)^{-1} \tag{9}$$

Obviously, if the new patch is the best one, $q_w > 1$ holds, otherwise $q_w \in (0,1]$. In the first case there will always be a waggle dance propagating this patch. Otherwise it depends on how the bee chooses randomly $p \in [0,1]$ and then if $p < q_w$ holds. The same evaluation happens when a bee has to decide on giving up a patch or after watching a waggle dance (see fig. 3).
Of course, on the implementation side, we have to keep track which vector index

$i \in \{1, 2, \ldots, n_w\}$ belongs to which path element. The path a bee chooses is copied to the bee's memory.

A path for a certain patch can get smaller by optimization (see chapter 4.4.1), thus the patch can obtain a better relative quality q_w. This will lead to higher propagation probability for the same patch. This means more bees will (over time) take the shorter path to any given flower patch, either by optimizing themselves or by looking at waggle dances of patches with shorter paths.

The question arises how bees can have such deep knowledge of the own patch and the quality of patches represented by waggle dances. *T.D. Seeley* [2] gives insight and assumptions on this matter. In chapter 5.4., p. 92, he states that bees can use dance duration, frequency and variation to communicate patch profitability. This indicates our model is an useful approximation in terms of waggle dance/path sharing. In chapter 5.5, he writes that travel distance and expected profit are the driving factors in determining patch quality. We also added the patch size A_w in our model, which should be fine as it is also a profit indicator when looking at the patch in a larger scale. The last question is if bees have preprogrammed knowledge what a good and a bad flower patch is or if this is a relative question depending on daily situations. We assumed the second one, and *T.D. Seeley* confirms this in chapter 5.7. on p. 104: For food unloading, hive bees prefer to unload the most profitable income first. This means bees with a bad flower patch have to wait longer, telling them their choice was probably not the best. Waiting times may vary randomly, thus adding randomness to the evaluations ($p < q$, fig. 3) makes sense as well.

3.3.3 Food collection

With every flight, a bee can gather $f_b = \min(f_{p,t}, f_{b,max})$ grams of pollen and nectar, where $f_{b,max}$ is the physical top limit of the bee. $f_{p,t}$ is derived from $f_{b,f}$ (see chapter 3.4.2) by multiplying $\max(f_{b,f})$ with the growing quality of a certain patch on day t, denoted $b_w \in (0, 1]$. Note that b_w itself is derived from the map (see chapter 4.2) and $f_{b,f}$. The daily food sum is increased every time a forager bee reaches the hive:

$$f_{d,t} = f_{d,t} + f_b \cdot F_{b,t} = f_{d,t} + \min(\max(f_{b,f}) \cdot b_w, f_{b,max}) \cdot F_{b,t} \tag{10}$$

An additional factor $F_{b,t}$ is used for clustering. This means we only simulate up to $\frac{F}{F_{b,t}}$ actual agents on day t and then interpolate to the actual value, which saves a lot of computation time. For relatively small cluster sizes, respectively many actual agents (in the order of 1000), the variation of $f_{d,t}$ will be small because the mean value of bees visiting a patch over a whole day is about the same as without

clustering, according to the law of large numbers (LLN).
Finally, we obtain a new equation to replace the original equation 6 for the daily food change:

$$\frac{df}{dt} = f_{d,t} - \gamma_A(F + H) - \gamma_B B \tag{11}$$

Where $f_{d,t}$ denotes the amount of food the foragers collected on day t.

We discuss the numerical results of this model in chapter 5.3.

3.4 Empirical data bases

3.4.1 Area around the hive

By analysing the different waggle dances of the forager bees, *T.D. Seeley* states in *Wisdom of the Hive* pp. 37ff that 95% of the foraging process occurs in a radius of 6.0 km (maximum distance at 10.9 km, mean distance at 2.2 km and the modal distance at 1.6 km [2]. The approximation of our modelled square $(6 \text{ km} \cdot 2)^2 \approx 100 \text{ km}^2$ should be sufficient for the simulation and comprehend all important data. In our model, we split the area up into a grid of 10 m $\cdot$ 10 m tiles of flower patches.

3.4.2 Flower patches and food

The flower patches themselves are divided into their different blossom: spring, summer and autumn. The empirical data was taken from *Wisdom of the Hive* p. 44f (year 1982). We merged the raw data and received a new graph for the whole year (see fig. 4). With equation 12, we can find out the weight of food a bee carries per flight.

$$f_{b,f}\,[\frac{g}{flight\,and\,bee}] = \frac{M\,[\frac{kg}{day}]}{C\,[bees] \cdot l\,[\frac{flights}{day\,and\,bee}]} \cdot 1000\,[\frac{g}{kg}] \tag{12}$$

The result $f_{b,f}$ is an approximation in gram per bee and flight. M denotes the change of the mass of the hive per day in kilograms (see fig. 4) and C represents the forager bee count approximated for a specific day of *T.D. Seeley's* empirical data [2]. l is the flight count a bee can do in average per day. Thus, we can take hive weight change per day as an indicator for the flower quality and flower type present at a given day of the year. This is the best approximation we tried, we did not find actual data of blooming and foraging profit as it is very difficult to measure this in nature.

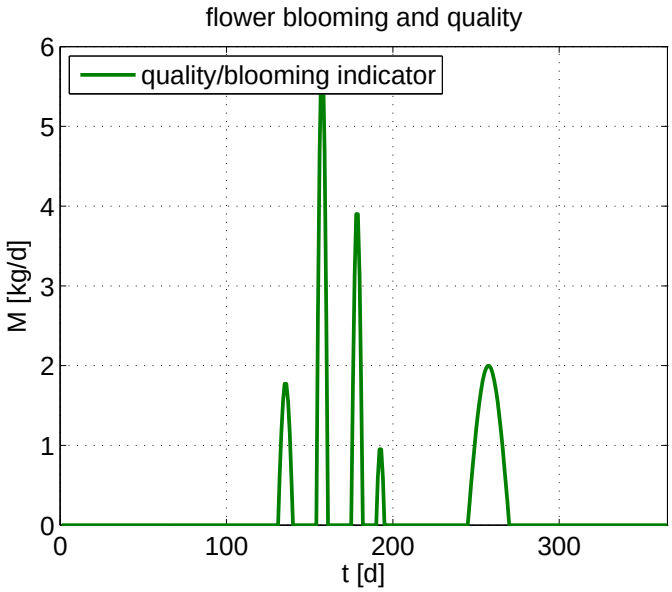


Figure 4: *This figure shows the change of the mass of the hive for any day in the year. It correlates with the quality of a flower patch and the type of flower, respectively.*

3.4.3 Bees' parameter

Our bee numbers are fully built on *D. S. Khoury's* model [1]. He used 16 000 hive bees and 8 000 forager bees which seems to be the right order of magnitude and is congruent with *T. D. Seeley's* estimation of about 20 000 [2].

v is the parameter that governs the effect of hive bees on brood survival, f governs the food effect on brood survival respectively (see section 3.1). $f = 50\ 000$ corresponds to 50 000 g of food to start with. The parameter $\varphi = \frac{1}{9}$, the adult bee emerging factor is assumed. Those values are extracted from *Khoury's* paper and are empirically tested but not explained any further.
Although, the food consumption for adult bees (0.007 g) and brood (0.018 g) is taken from explicitly mentioned data [9].

As remarked at the beginning of section 3.1, we simplify the complicated ageing process (larvae to hive bee) to 12 days. Mortality rate is set to 0.075 and thus slightly lower than the values of *Khoury*. This is discussed later in section 5.2.
The recruitment factor $\alpha = 0.25$ is set according to *Khoury's* model.

4 Implementation

4.1 General implementation

The basic simulation consists only of evaluating equations for every time step. The implementation can be found in the *Hive.m* code. Variable names are chosen so that they match *Khoury et al.* [1] and chapter 3.1 as close as possible.
For the advanced model, the following hierarchy is used: The top level is the *hive_simulation.m* file, which sets up the whole simulation. Then it keeps track of data and iterates through days in the hive simulation and through seconds in the environment simulation. The *Prop* class (*Properties_Base.m*) is an implicit struct which allows to plug in and change empirical data and simulation parameters. For the environment simulation, *Bee* (*Bee.m*) is the class for agents, *Hive* (*Hive.m*) for hives and *World* (*World.m*) for the maps.
The final report (simulation results) is stored as class *Report* (*Report.m*) with an implicit struct, making data analysis easy and extendible.

4.2 Map and flower patch quality

The bees all navigate on a map with the hive in the middle of the map. Every pixel on the map equals to a patch of 100 m^2.

The map is encoded in HSV (fig. 5). From the 360° color hue circle ($H_{HSV} \in \{0, 1, \ldots, 360\}$), always 60° are assigned to one integer type value $j \in \{0, \ldots, 5\}$. Value 0 is used for empty space and values 1 to 3 for the three flower seasons/types we implemented. Values 4 and 5 are for future extensions such as smog/pesticide sources. More values can be gained by assigning less than 60° per type. The saturation $S_{HSV} \in [0, 1]$ is always kept at 1. For the flower patch quality basis, the $V_{HSV} \in [0, 1]$ component is used.

Figure 5: *Map encoding in HSV.*

To obtain the daily map for environment simulation, the normalized blooming

indicator (see fig. 4) of each flower type $i \in \{1, 2, 3\}$ is separately multiplied with the V_{HSV} value of the basis map where flower type i occurs (see fig. 6). As a result, some flowers don't appear on the map and those that appear get scaled in quality according to the daily situation (see fig. 2). From the obtained scaled quality map, the bees now read the value b_w when visiting a patch. This is the driving factor for forager distribution (see chapter 3.3.2).

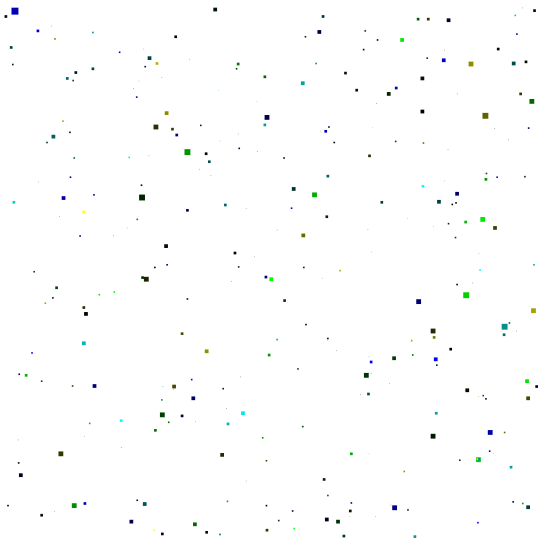


Figure 6: *Example of an equal distributed flower patch, normal distributed quality map. Note that the encoding is given by Figure 5 but background black has been swapped out with white for better visibility.*

4.3 Scouts

4.3.1 Random walk

Very little is known about the searching behaviour of scout bees [2], p. 87f. *T.D. Seeley* mentions that scouts fly low above the ground and inspect flowers they encounter during their flight. Naturally, the flight distance is limited. Considering this, a random walk is the most logical behaviour to simulate scout bees. The random walk is stopped when a scout encounters a new flower. Previously discovered flower patches are ignored by scouts, as bees mark already visited patches with the Nasonov pheromone [10] p. 133ff.
The random walk is preformed in following steps:

- The bee starts from the hive with an equally distributed random angle $\alpha \in [0, 2 \cdot \pi]$ (radian).

- After a number of time steps (we've chosen one time step, so 60 seconds), the angle α is changed by at most $r \cdot 0.5$ (radical angle changes are not desired), where r is a normal distributed random value.
- Between two angle changes, the scout bee flies with 7 m/s into the chosen direction.
- The path points are recorded at every angle change point. The distance passed is calculated and stored in a L^2 norm scalar.

The path a scout bee walks is recorded in a vector of x and y coordinates:

$$\begin{pmatrix} x_0 & x_1 & \dots & x_n \\ y_0 & y_1 & \dots & y_n \end{pmatrix}$$

where $\begin{pmatrix} x_0 \\ y_0 \end{pmatrix}$ are the hive coordinates and $\begin{pmatrix} x_n \\ y_n \end{pmatrix}$ are either the coordinates with maximum possible distance from the hive according to L^2 norm, or the coordinates of a flower patch.

An example of such a random walk is given in Figure 7.

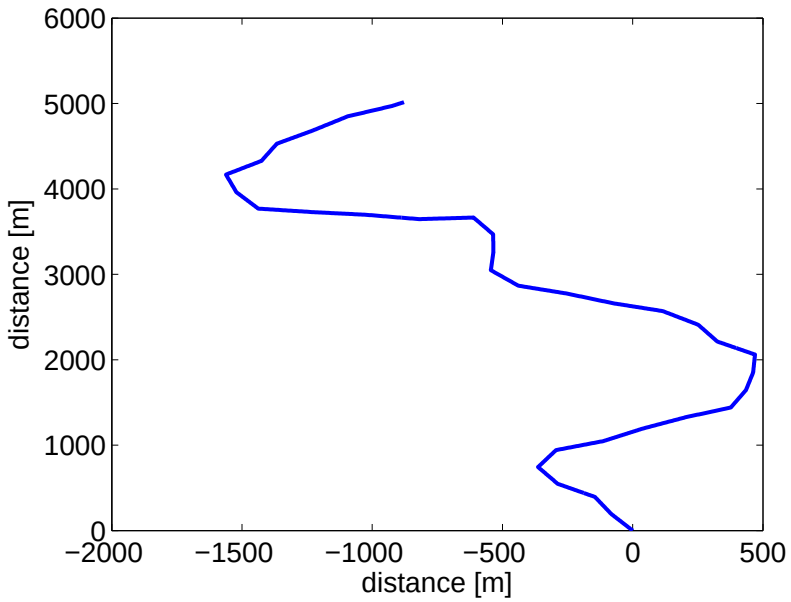


Figure 7: *Example of a random walk executed by a scout bee.*

4.3.2 Bresenham line algorithm

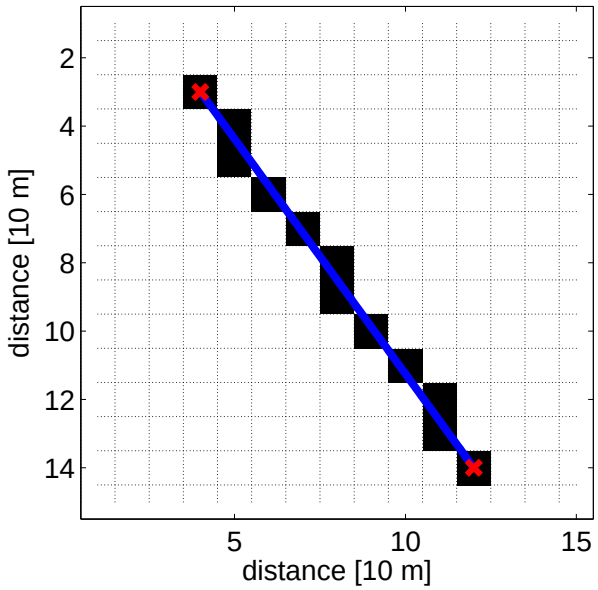


Figure 8: *Example of Bresenham's line algorithm with map and path segment.*

As the scout bees pass a distance of $v \cdot \Delta t_s$, respectively $\sqrt{(v_x \cdot \Delta t_s)^2 + (v_y \cdot \Delta t_s)^2}$ for a velocity v and time step Δt_s in every iteration step, multiple field are being passed at once. In order to check all fields between the last and current iteration step for flower patches, the Bresenham line algorithm is used. With this method, all integer coordinates between two given points on the map are reported back. Afterwards, every obtained point can be checked against the current quality- and type-map of the environment simulation. For this simple algorithm, a pre-existing implementation is used [11]. Figure 8 is an example on a 15x15 map with coordinates $(x = 4, y = 3)$ to $(x = 12, y = 14)$ and reported points (black boxes).

4.4 Foragers

4.4.1 Path optimization

We assume bees can optimize the original path they obtain from a waggle dance, as the bees are able to orientate themselves in the environment with sun positioning [2], p. 37. The implementation used is a simple one, in order to keep computation times low. At every point where optimization may occur (see chapter 3.3.1), there

is a 50% chance of optimization. Optimization works as follows:

$$\begin{pmatrix} x_0 & x_1 & x_2 & x_3 & x_4 & \dots & x_{n-3} & x_{n-2} & x_{n-1} & x_n \\ y_0 & y_1 & y_2 & y_3 & y_4 & \dots & y_{n-3} & y_{n-2} & y_{n-1} & y_n \end{pmatrix}$$

$$\Longrightarrow_{optimization} \begin{pmatrix} x_0 & x_2 & x_4 & \dots & x_{n-4} & x_{n-2} & x_n \\ y_0 & y_2 & y_4 & \dots & y_{n-4} & y_{n-2} & y_n \end{pmatrix}$$

This means every second way point is skipped, while starting- and endpoints are preserved. According to the triangle inequality, the L^2 norm of the distance can only become smaller after every such step, therefore it is an optimization in terms of path length. The outcome of such an optimization process on the path from Figure 7 is presented in Figure 9.

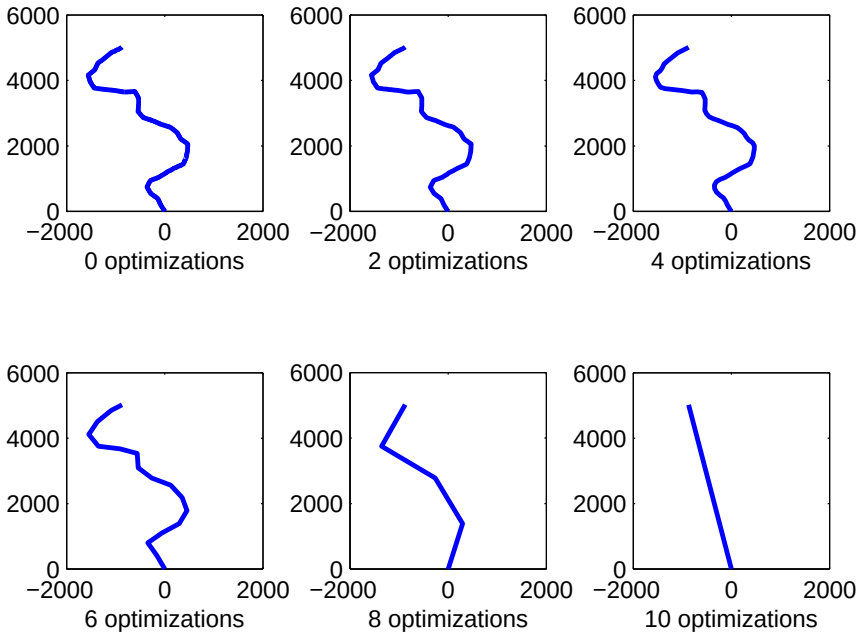


Figure 9: *Example of path optimization used to short cut the path to flower patches.*

Comparing this model of optimization with empirical observations made by *Lihoreau et. al* [12], p. 3, shows that our model is a good approximation: They state

that the bees reduce their travel times gradually with experience and higher numbers of flights. The same happens with our model, as the shortest path is not taken directly, but shorter paths are taken after visiting a certain patch often enough. The only difference is that we don't explicitly simulate multi-patch sequences (bees can visit multiple patches before returning to the hive) while *Lihoreau et. al* primarily focus on this matter.

5 Simulation Results and Discussion

5.1 Constant food and egg laying

The first iterative simulation run is based only on the equations of *David S. Khoury et al.*[1]. As expected the hive stabilizes at an equilibrium point while the stored food increases infinite. (See graph A.1.2)

5.2 Constant food, dynamic egg laying

The second iterative simulation run was to see if the new laying function (Figure 1) was working as intended. As a result the bee population does not stabilize at one point, it now describes a more natural periodic form with a population maximum around August and a minimum around February.

The colony we simulated did not survive with any daily mortality rates higher than 0.1. There are not enough bees emerging from eggs in order to compensate the mortality rate. However, we found that the new mortality rates assumed in our simulation (0.075 daily mortality) are still realistic. *Henry et al.* [13] describe mortality rates between 0.102 and 0.316 in empirical testing, which is the range in which *Khoury et al.* [1] simulated. *R. Dukas* [14] observed that mortality of foragers is in this range mainly because of predation. As we apply the mortality to all days, even in times and seasons without foraging (all bees stay in the hive), our simulated mortality rates are naturally lower.

The stored food still increases infinite but now the correlation between the number of forager bees and the daily increase in stored food is very to see (linear correlation coefficient = 0.9775).

5.3 Environmental model

For testing the environmental model, we decided to change the quality indicator and delay seasons (especially autumn). The map (see chapter 4.2) is always a randomly

generated map with equally distributed flower patches across the map and normal distributed flower patch quality. All other parameters are kept fixed so that the results are unambiguous.

5.4 Empirical data based run

In this run we wanted to see if our model can produce the same results as *T.D. Seeley's* experiments from *Wisdom of the hive* pp. 44 fig. 2.14. Our date correlates quite well with the empirical data before swarming occurs, year 1982 compared to graph A.2.2. As swarming was never a part of our model were satisfied with the accuracy of our model. Compared to our earlier simulations, the food collection rate is now dominated by the availability of flowers. The forager count is still a important factor as later simulations will show.

5.4.1 Typical simulated day

In a typical summer day after 2.5 houres 80"%" of the patches are discovered by the scout bees.

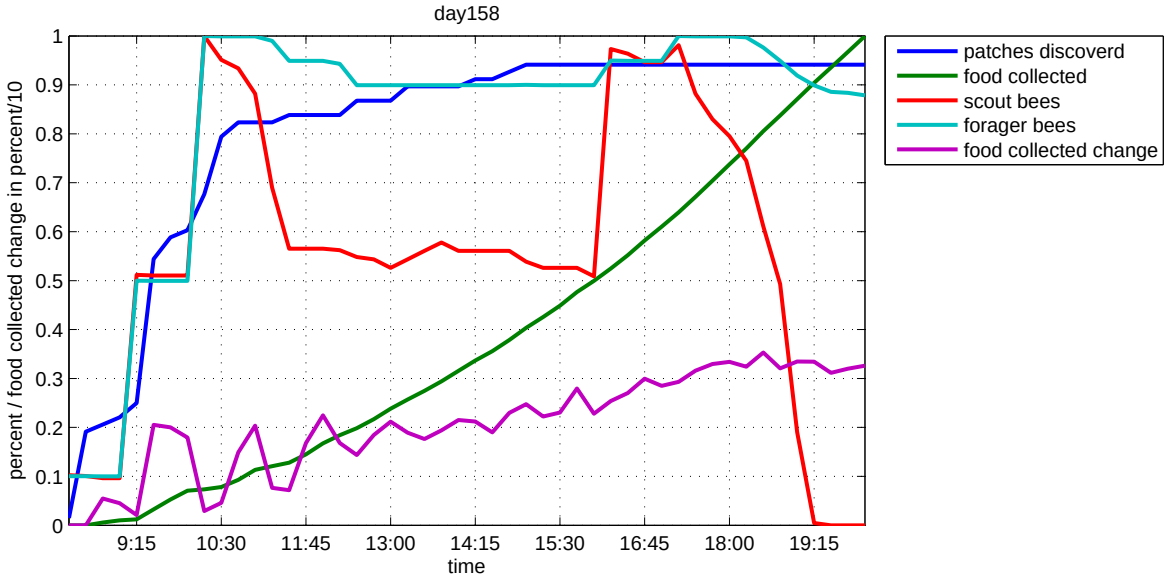


Figure 10: *A summer day from the standard run R1_1*

5.4.2 Missing flower seasons

5.4.3 Variations on autumn flowers

5.4.4 Model restrictions

Even though we extended *Khoury's* model by environmental influences on flowers and food reward, it is far from complete. One big divergence from nature is the treatment of pollen and nectar as one. Pollen, which is the protein source, and nectar, which is the carbohydrate source, are actually collected by two different forager classes intra-colonially [15].
Another big factor of disturbance are diseases and infections. Impairments in different aspects of a bee's life and behaviour, caused by pesticides or genetic mutation is entirely left out. There are also no other environmental influences such as aridity or wetland, human or invasive impacts of other species.
Regarding the simulation of a bee's way of "thinking", we might never find a satisfying answer, it is only a model.

6 Summary and Outlook

7 References

[1] D. S. Khoury, B. A. Barron, and R. M. Myerscough. "Modelling Food and Population Dynamics in Honey Bee Colonies". In: *PLOS ONE* (2013).

[2] T. D. Seeley. *The Wisdom of the Hive. The Social Physiology of Honey Bee Colonies.* London: Harvard University Press, 1995.

[3] Simon G. Potts et al. "Global pollinator declines: trends, impacts and drivers". In: *Trends in Ecology & Evolution* 25.6 (2010), pp. 345–353. ISSN: 0169-5347. DOI: 10.1016/j.tree.2010.01.007.

[4] Michel Thomann et al. "Flowering plants under global pollinator decline". In: *Trends in Plant Science* 18.7 (2013), pp. 353–359. ISSN: 1360-1385. DOI: 10.1016/j.tplants.2013.04.002.

[5] Axel Decourtye et al. "Imidacloprid impairs memory and brain metabolism in the honeybee (Apis mellifera L.)" In: *Pesticide Biochemistry and Physiology* 57.3 (2004), pp. 410–419. ISSN: 0147-6513. DOI: 10.1016/j.ecoenv.2003.08.001.

[6] A. Bernadou et al. "Effect of fipronil on side-specific antennal tactile learning in the honeybee". In: *Journal of Insect Physiology* 55.12 (2009), pp. 1099–1106. ISSN: 0022-1910. DOI: 10.1016/j.jinsphys.2009.08.019.

[7] O. Schweiger et al. “Multiple stressors on biotic interactions: How climate change and alien species interact to affect pollination”. In: *Biological Reviews* 85.4 (2010), pp. 777–795.

[8] Walt Wright. *How Many Eggs CAN a Queen Lay?* 2008. URL: http://www.beesource.com/point-of-view/walt-wright/how-many-eggs-can-a-queen-lay/.

[9] John R. Harbo. “Effect of brood rearing on honey consumption and the survival of worker honey bees.” In: *Journal of Apicultural Research* 32 (1993), pp. 11–17.

[10] Mark L. Winston. *The Biology of the Honey Bee*. London: Harvard University Press, 1991.

[11] Peter Corke. *Robotics and Machine Vision Toolboxes for MATLAB (MVTB)*. URL: https://code.google.com/p/matlab-toolboxes-robotics-vision/.

[12] Steven C. Lecomber Mathieu Lihoreau Lars Chittka and Nigel E. Raine. “Bees do not use nearest-neighbour rules for optimization of multi-location routes”. In: *biology letters* (2011). DOI: 10.1098/rsbl.2011.0661.

[13] Henry et al. “A Common Pesticide Decreases Foraging Success and Survival in Honey bees”. In: *Sciencemag* (2012).

[14] R. Dukas. “Mortality rates of honey bees in the wild”. In: *Insectes Sociaux* (2008).

[15] Thomas Schmickl and Karl Crailsheim. “HoPoMo: A model of honeybee intracolonial population dynamics and resource management”. In: *Ecological Modelling* 204.1–2 (2007), pp. 219–245. ISSN: 0304-3800. DOI: 10.1016/j.ecolmodel.2007.01.001.

A Additional Graphics

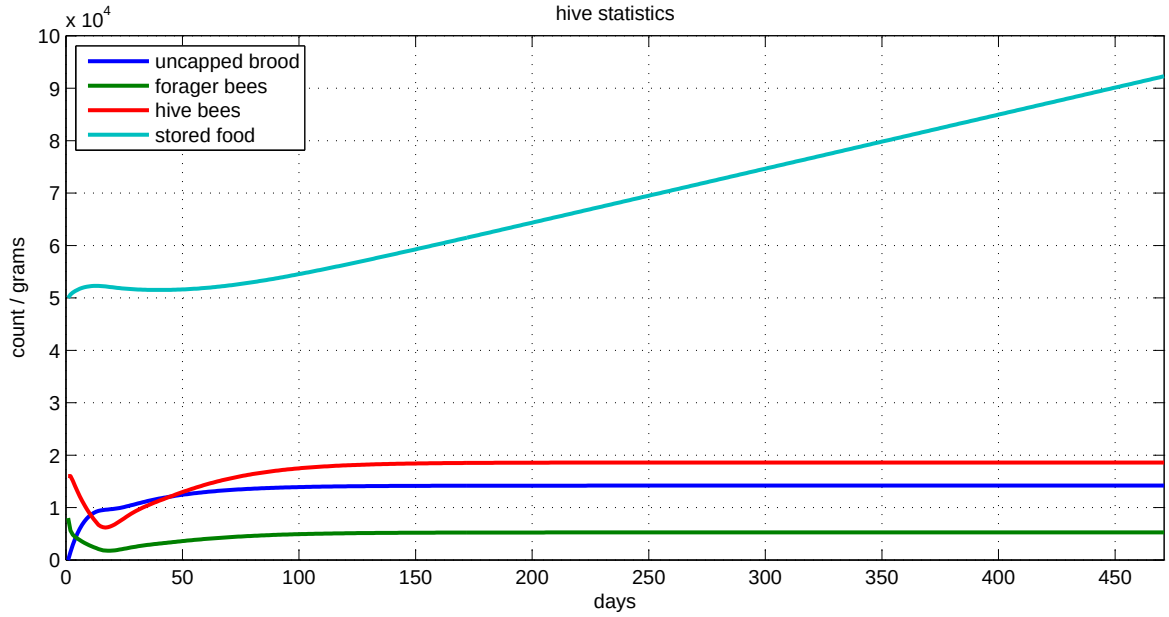
A.1 Simulation run without environment simulation

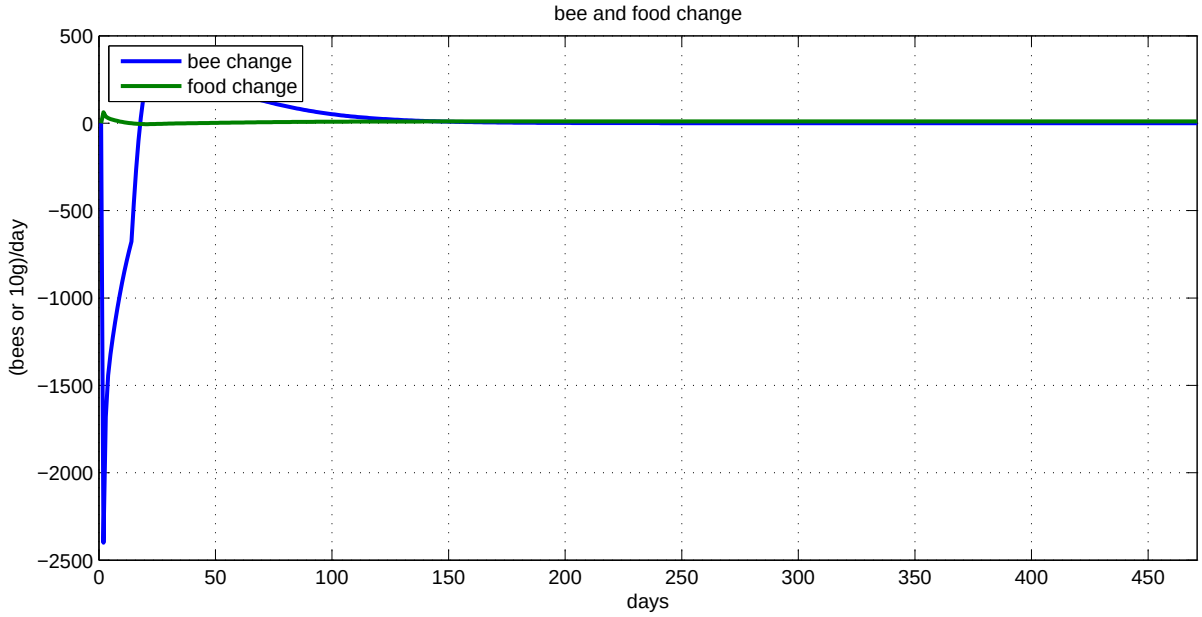
A.1.1 Runscript with changes relative to the base properties

```
% This is a runscript for automatized simulation on any system

% Simple test cases based on Properties_Base
run('data\Properties_Base.m');

% Load the properties into memory, make copies and modify them on the go

% The testcases:
% 1. Original model (constant food, constant laying rate, original mortality)
% 2. Intermediate model (constant food, varying laying rate, adapted mortality)

Prop1 = Prop;
```

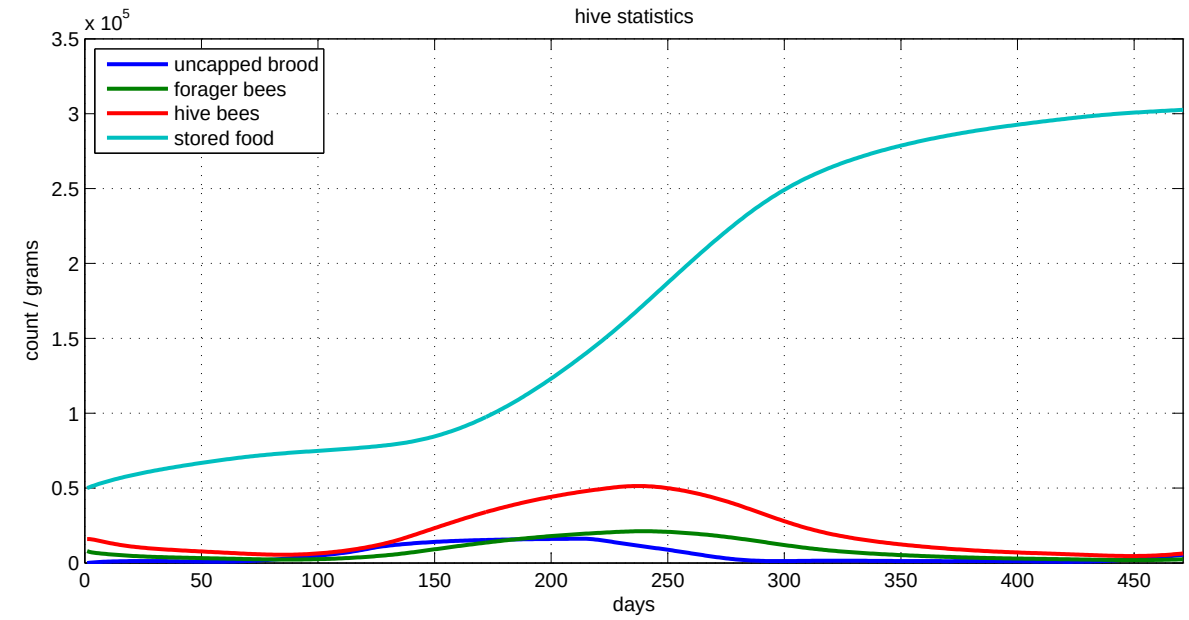
```
Prop1.Sim.Hive(1).fixed_food_rate = 1;
Prop1.Sim.Hive(1).mortality = 0.3;
Prop1.Sim.Hive(1).laying_function = [0,1,2;1,1,1];
Prop1.Sys.identifier = 'Properties_Base_R0_1';

Prop2 = Prop;
Prop2.Sim.Hive(1).fixed_food_rate = 1;
Prop2.Sys.identifier = 'Properties_Base_R0_2';


proparray=[Prop1,Prop2];

% Start simulation
hive_simulation(proparray);
```

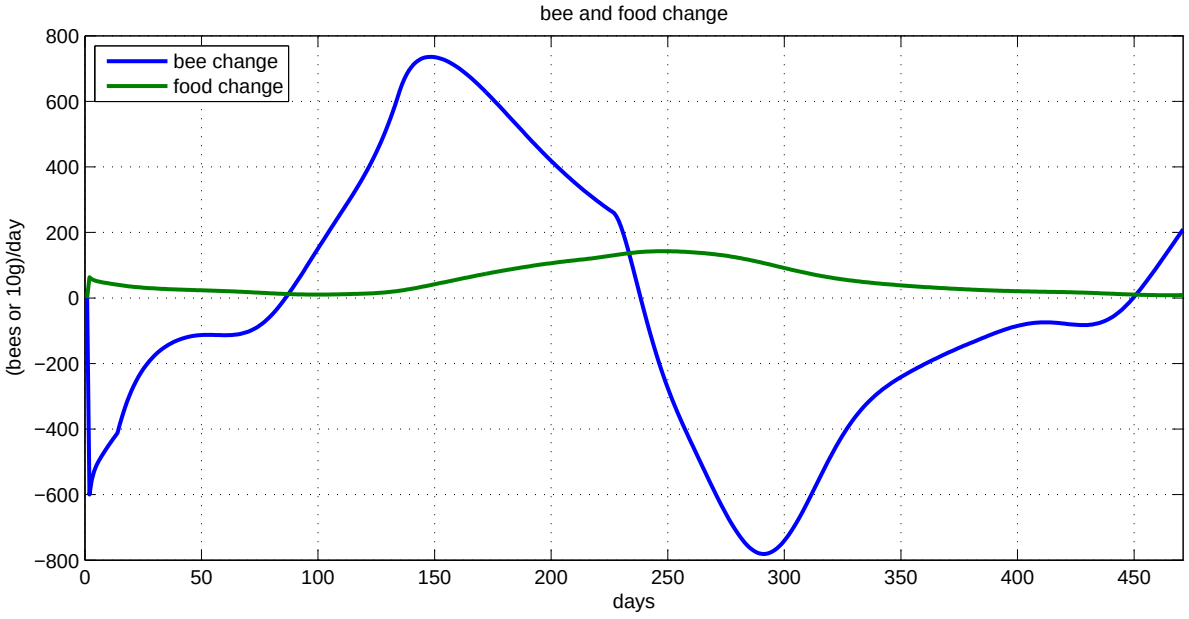
A.1.2 Constant laying rate, constant food income





A.1.3 Varying laying rate, constant food income



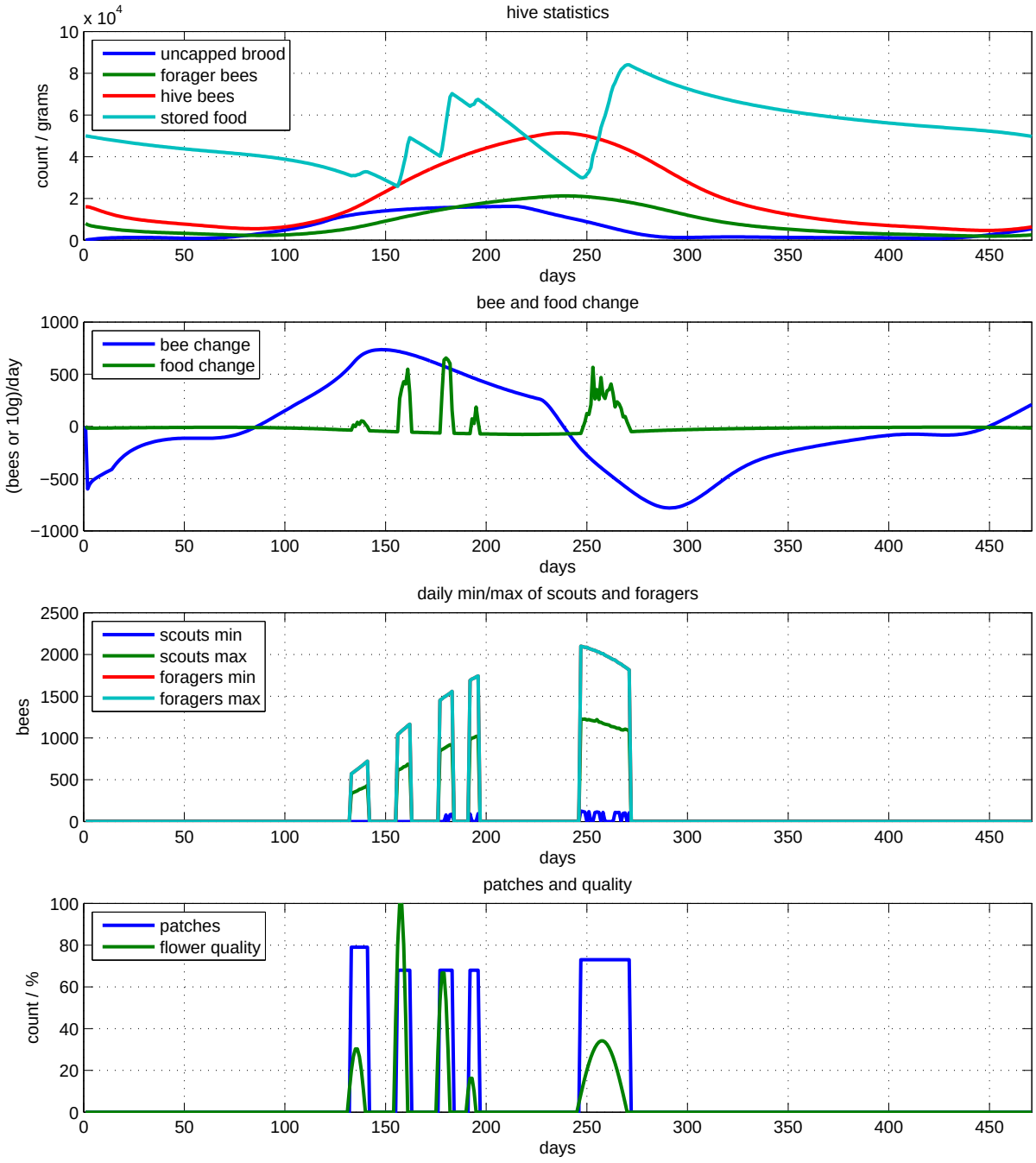


A.2 Simulation run with missing flower patches

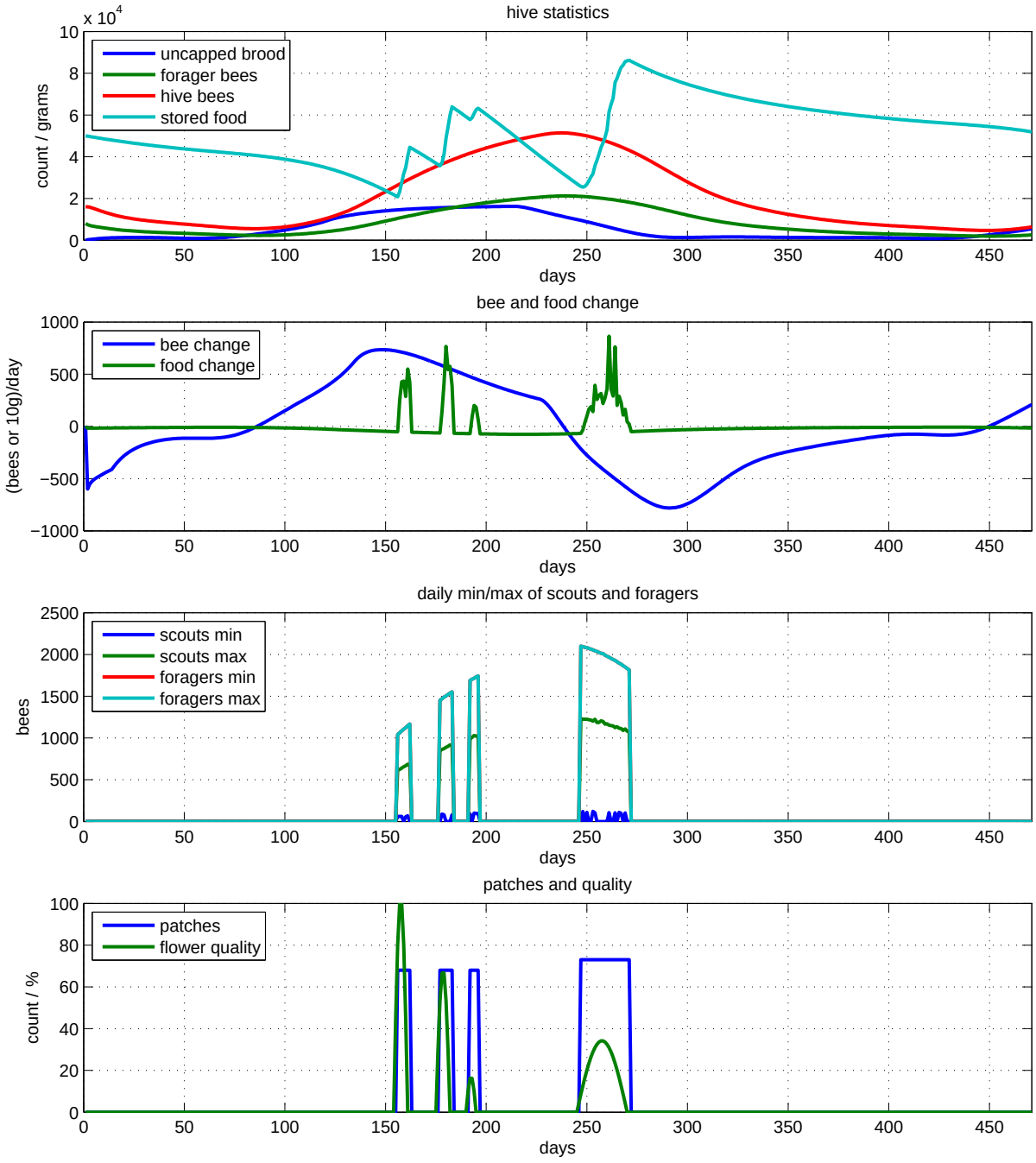
A.2.1 Runscript with changes relative to the base properties

```
% This is a runscript for automatized simulation on any system

% Simple test cases based on Properties_Base
run('data\Properties_Base.m');

% Load the properties into memory, make copies and modify them on the go

% The testcases:
% 1. all flowers
% 2. no spring flowers
% 3. no summer flowers
% 4. no autumn flowers

Prop1 = Prop;
Prop1.Sys.identifier = 'Properties_Base_R1_1';

Prop2 = Prop;
Prop2.Sim.Flower(1).year_activity = [1:2;0,0];
Prop2.Sys.identifier = 'Properties_Base_R1_2';

Prop3 = Prop;
Prop3.Sim.Flower(2).year_activity = [1:2;0,0];
Prop3.Sys.identifier = 'Properties_Base_R1_3';

Prop4 = Prop;
Prop4.Sim.Flower(3).year_activity = [1:2;0,0];
Prop4.Sys.identifier = 'Properties_Base_R1_4';


proparray=[Prop1,Prop2,Prop3,Prop4];

% Start simulation
hive_simulation(proparray);
```

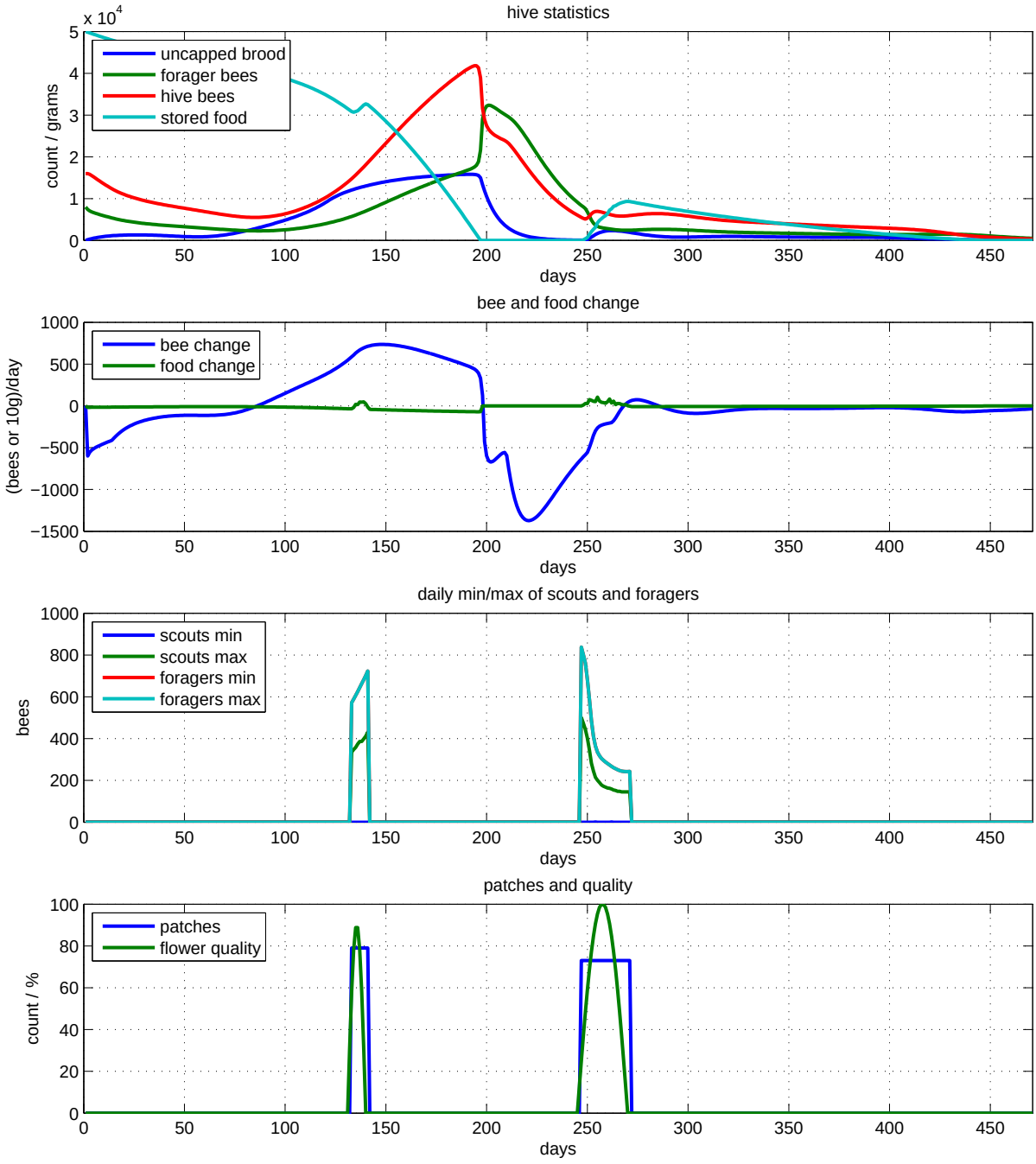
A.2.2 All flower patches



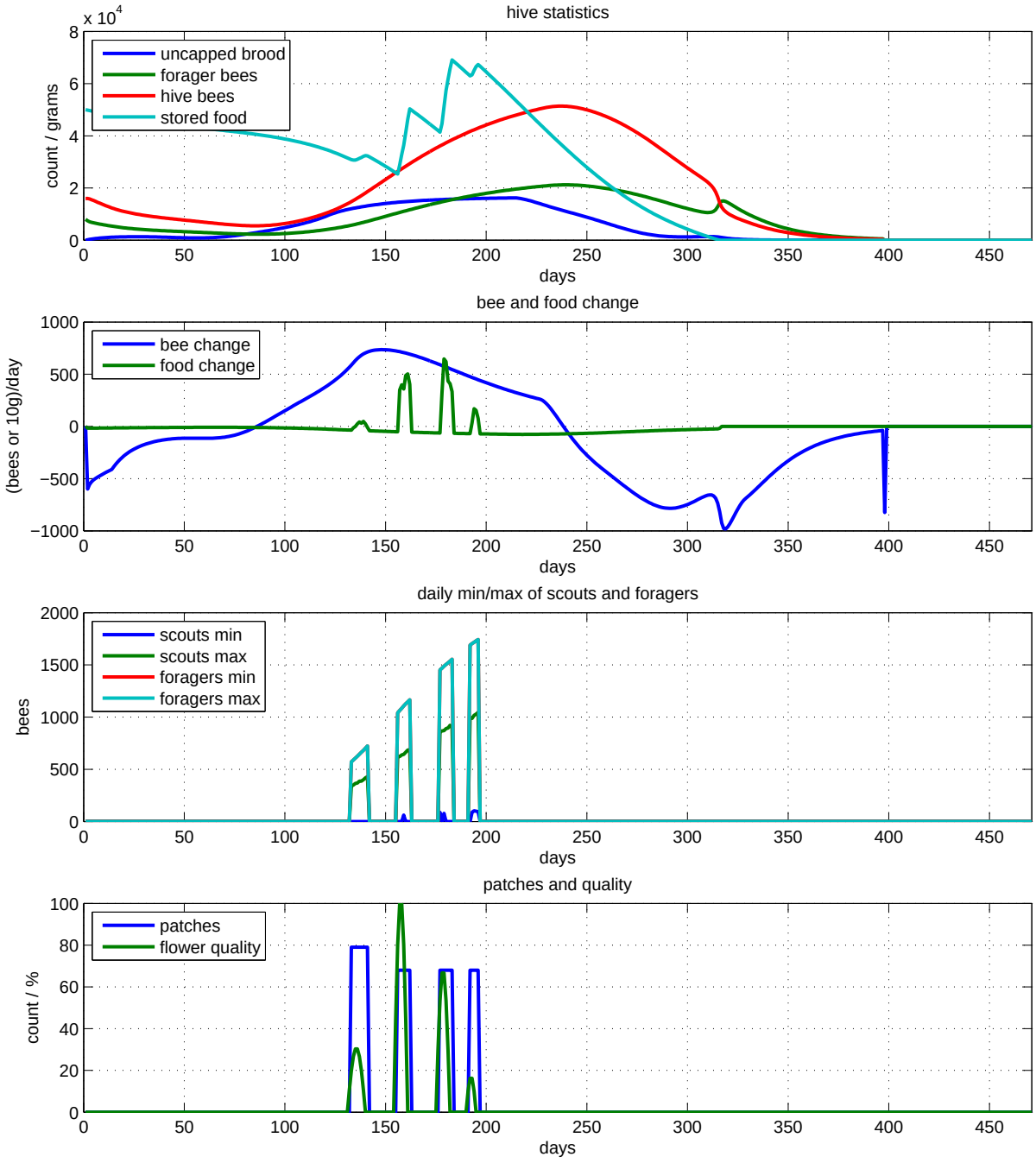
A.2.3 Missing spring flowers



A.2.4 Missing summer flowers



A.2.5 Missing autumn flowers

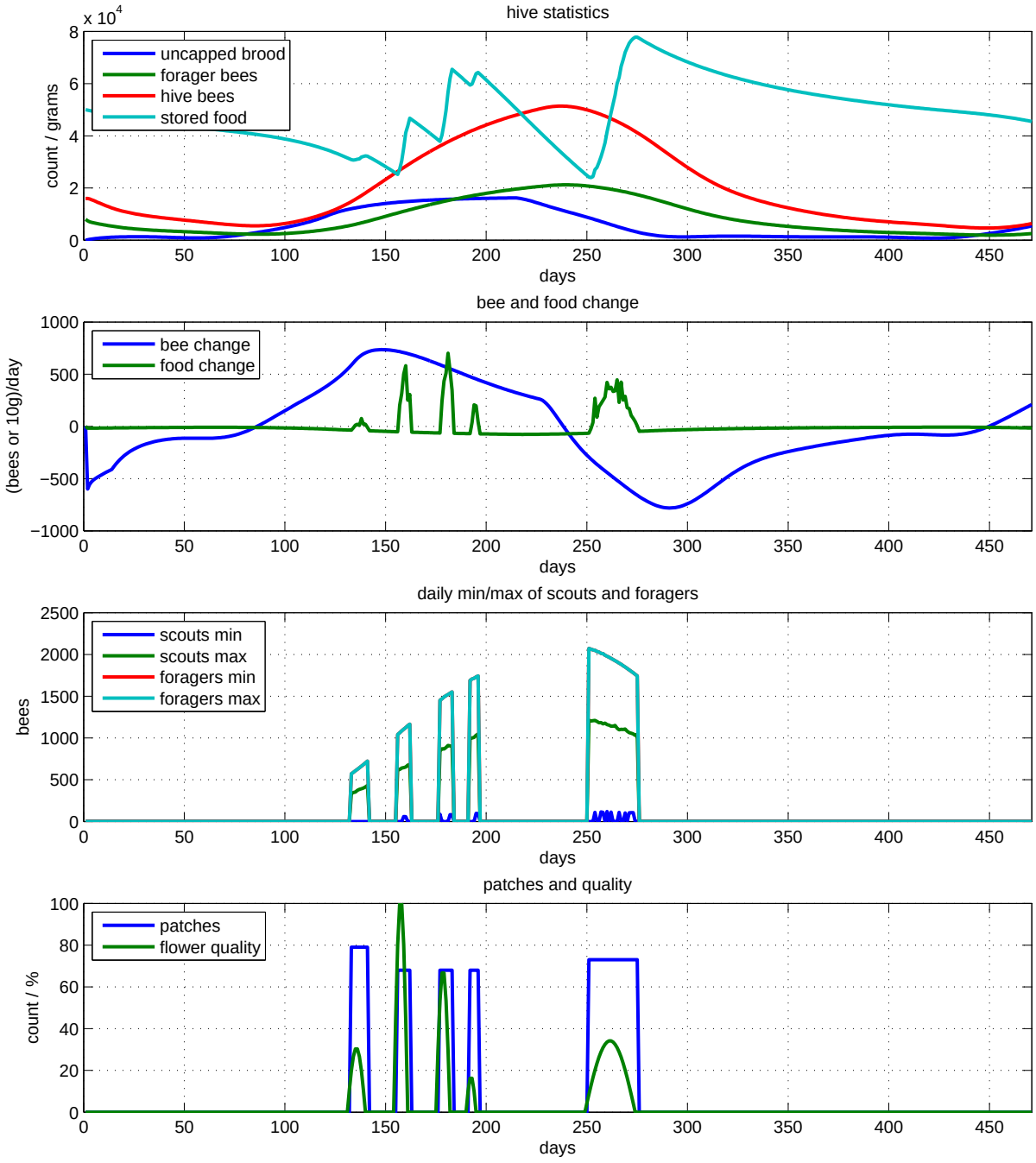


A.3 Simulation run with varying autumn flowers

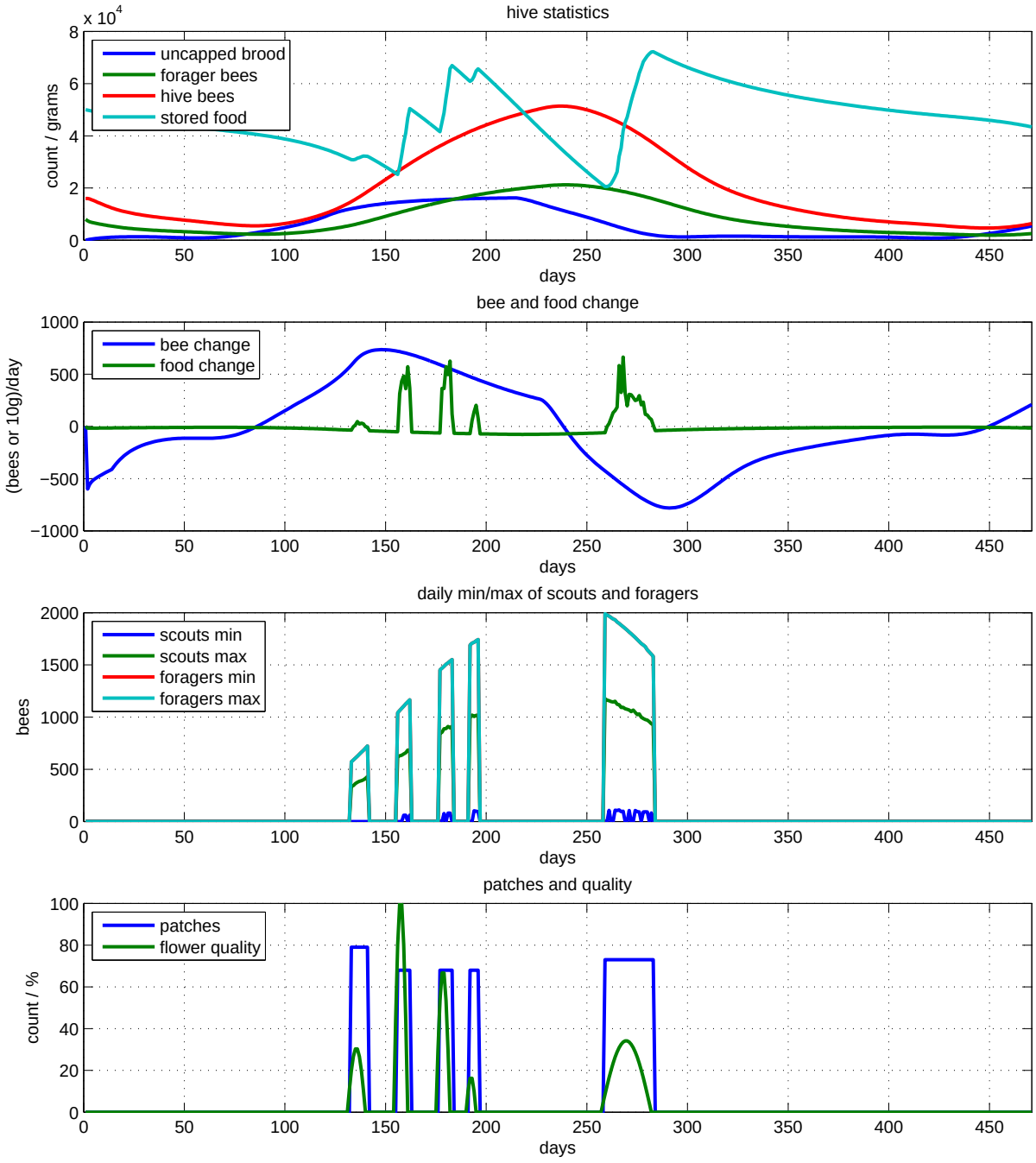
A.3.1 Runscript with changes relative to the base properties

```
% This is a runscript for automatized simulation on any system

% Simple test cases based on Properties_Base
run('data\Properties_Base.m');

% Load the properties into memory, make copies and modify them on the go

% The testcases, variable change 1:
% 1. Autumn flower peak = 2
% 2. Autumn flower peak = 1.5
% 3. Autumn flower peak = 1
% 4. Autumn flower peak = 0.5

% The testcases, variable change 2:
% 1. Autumn flowers shifted by 4 days
% 2. Autumn flowers shifted by 8 days
% 3. Autumn flowers shifted by 12 days
% 4. Autumn flowers shifted by 16 days
% 5. Autumn flowers shifted by 20 days

% Empty proparray at first
proparray = [];

% 4 possible peaks to test
peaks = [2,1.5,1,0.5];

for j=1:5
    for i=1:4
        identifier = strcat('Properties_Base_R2_',num2str(i),'_',num2str(j));
        tprop = Prop;
        tprop.Sys.identifier = identifier;
        tprop.Sim.Flower(3).year_activity(1,:) = tprop.Sim.Flower(3)
            .year_activity(1,:) + j * 4;
        tprop.Sim.Flower(3).peak = peaks(i);
        proparray = [proparray, tprop];
    end
end

% Start simulation
hive_simulation(proparray);
```

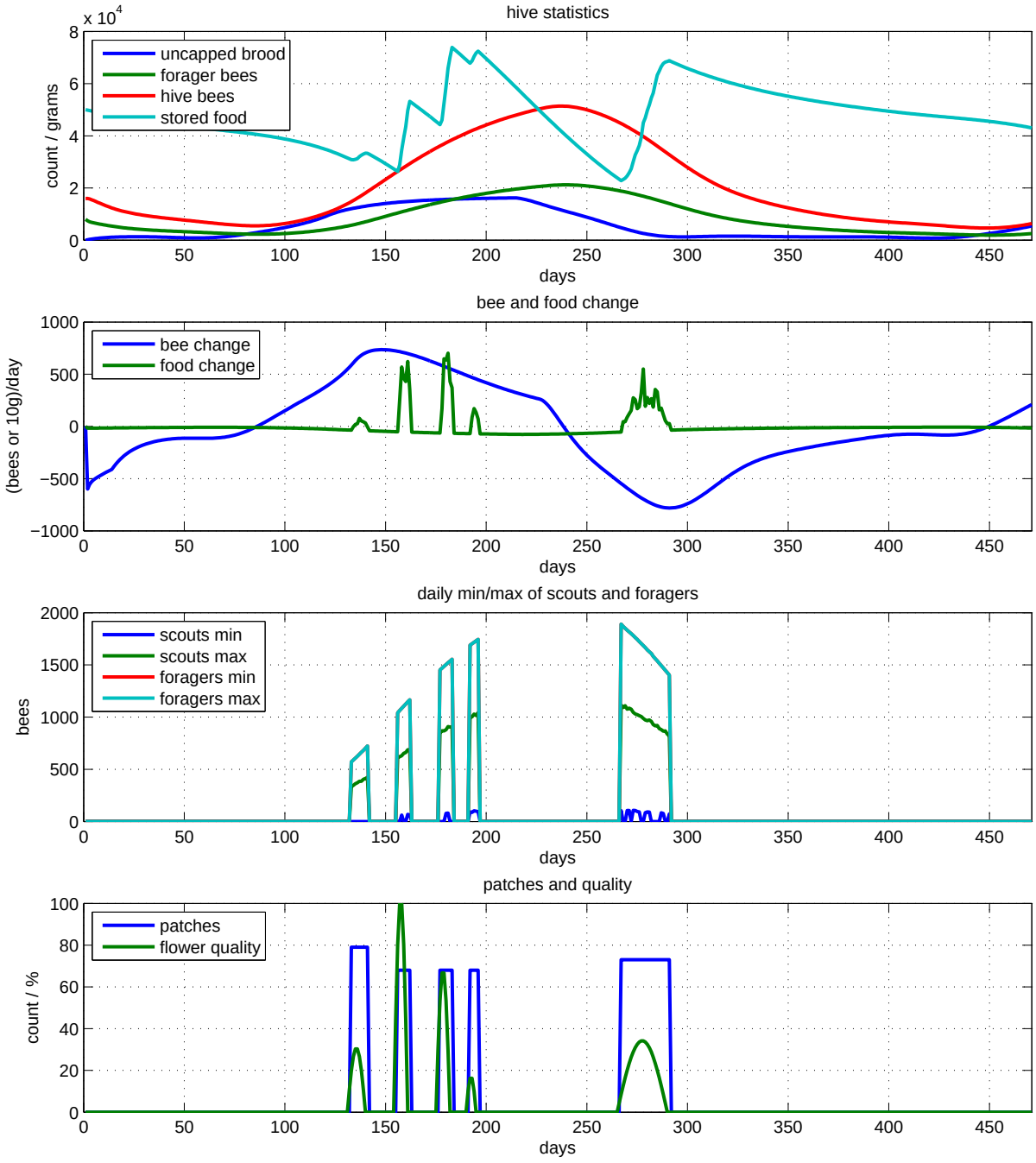
A.3.2 Peak = 2, delayed by 4 days



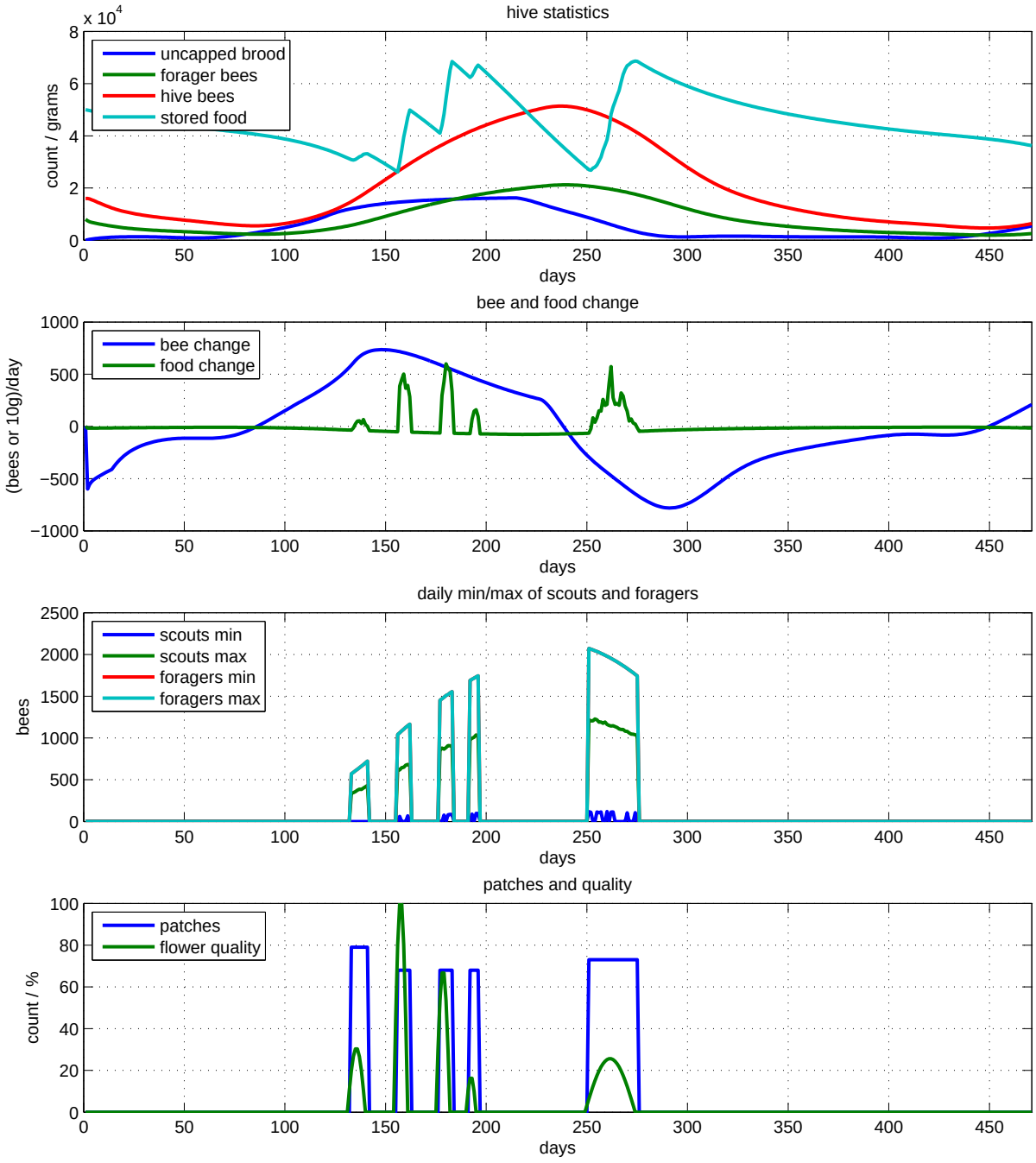
A.3.3 Peak = 2, delayed by 12 days



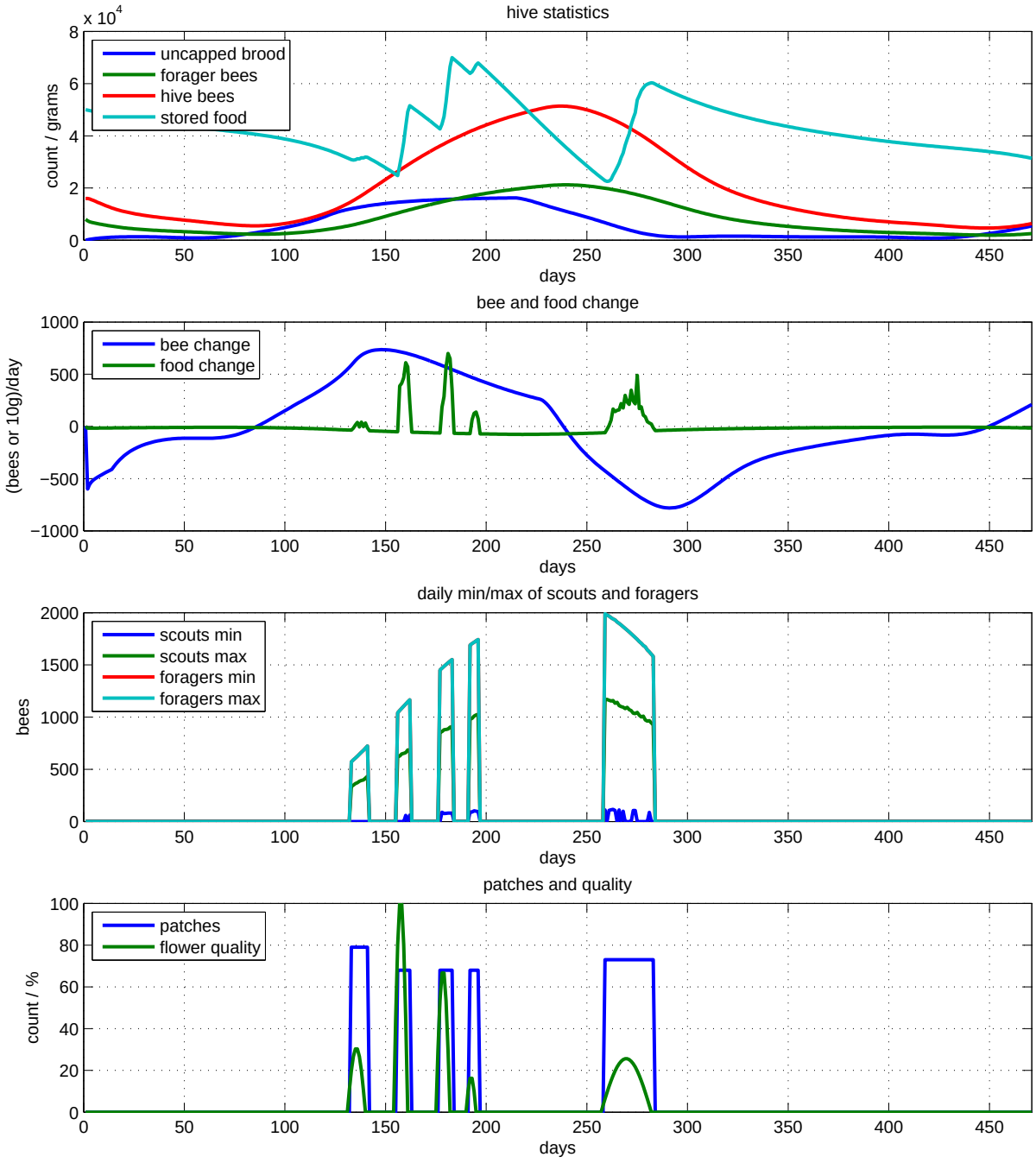
A.3.4 Peak = 2, delayed by 20 days



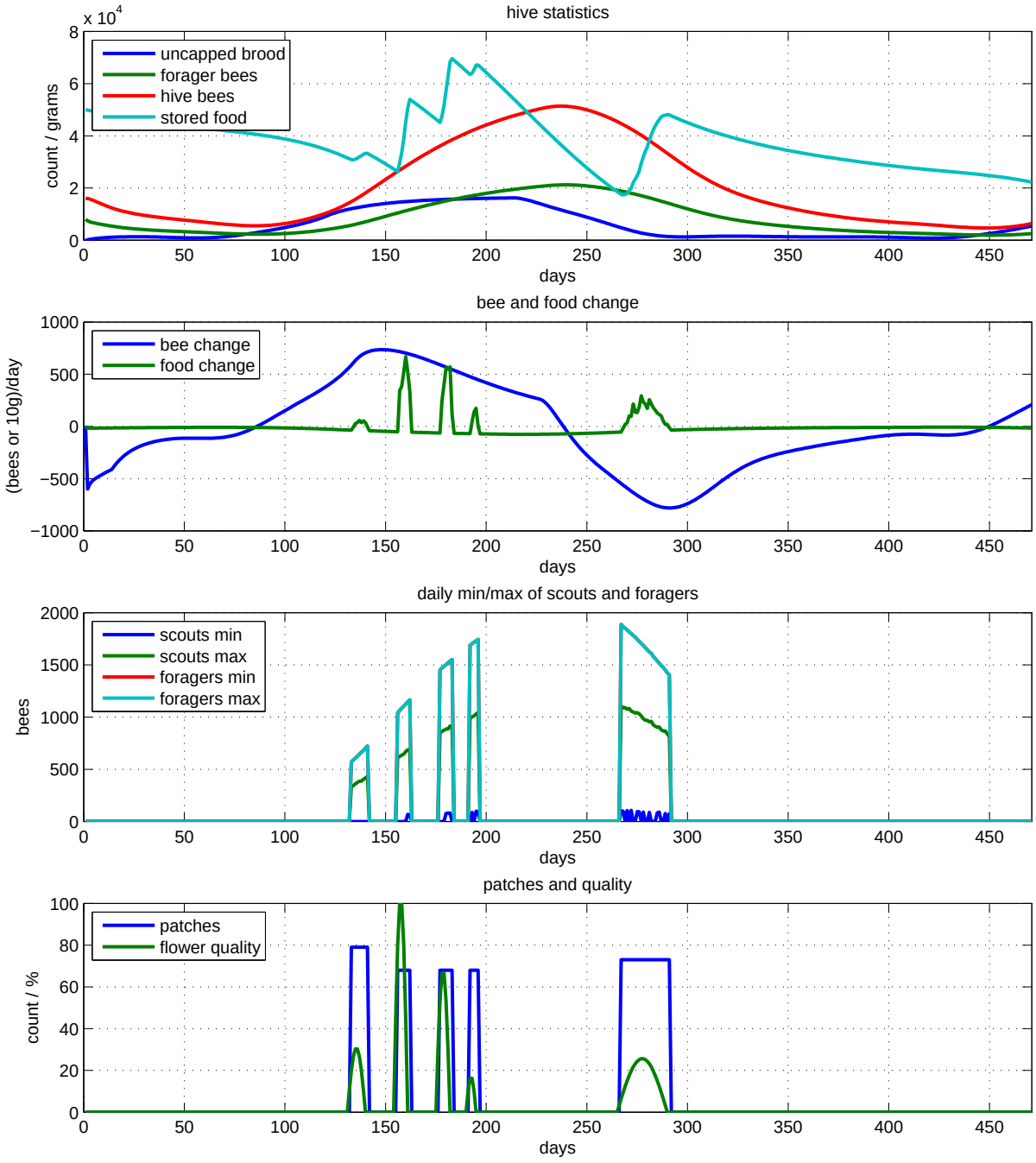
A.3.5 Peak = 1.5, delayed by 4 days



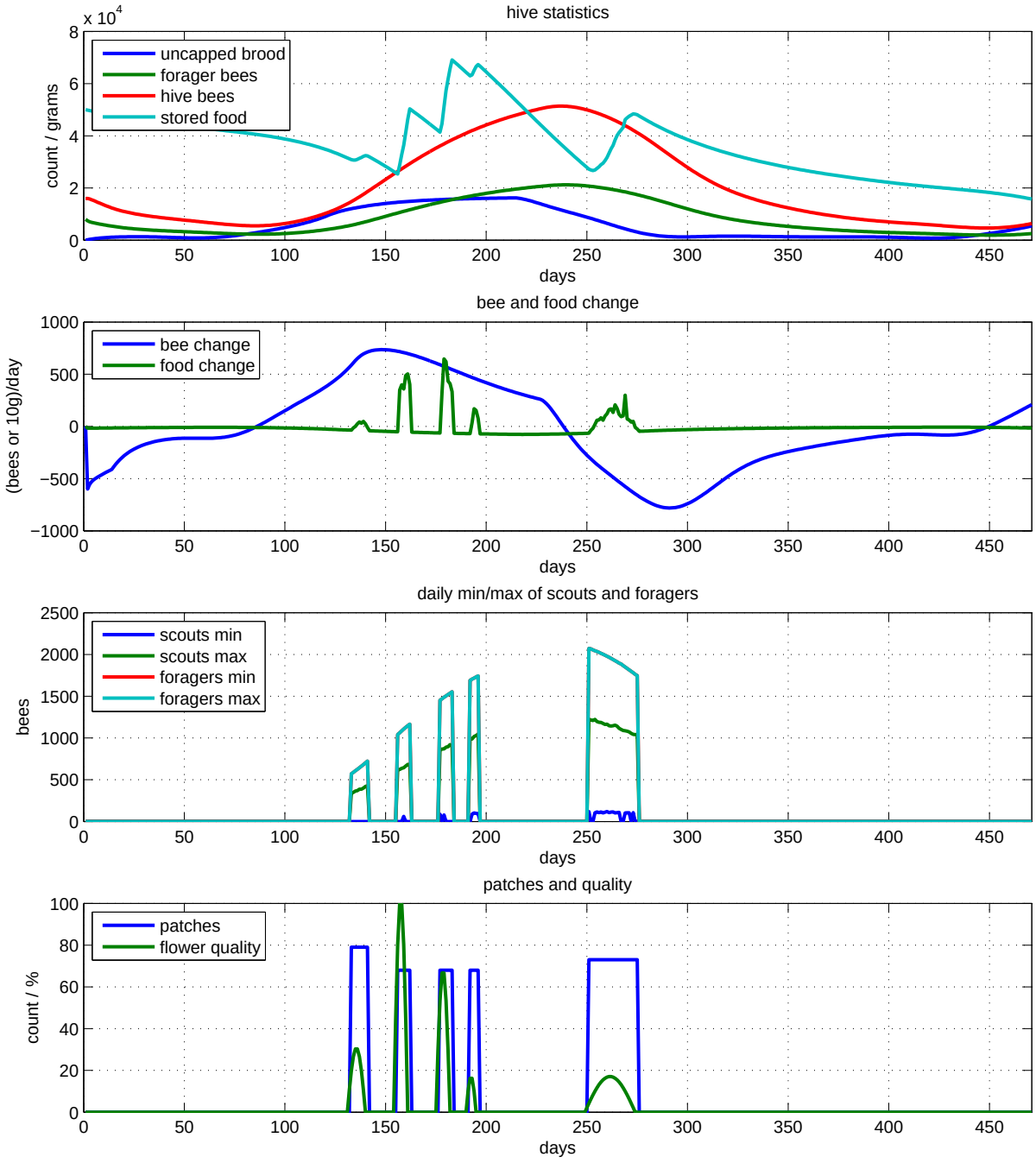
A.3.6 Peak = 1.5, delayed by 12 days



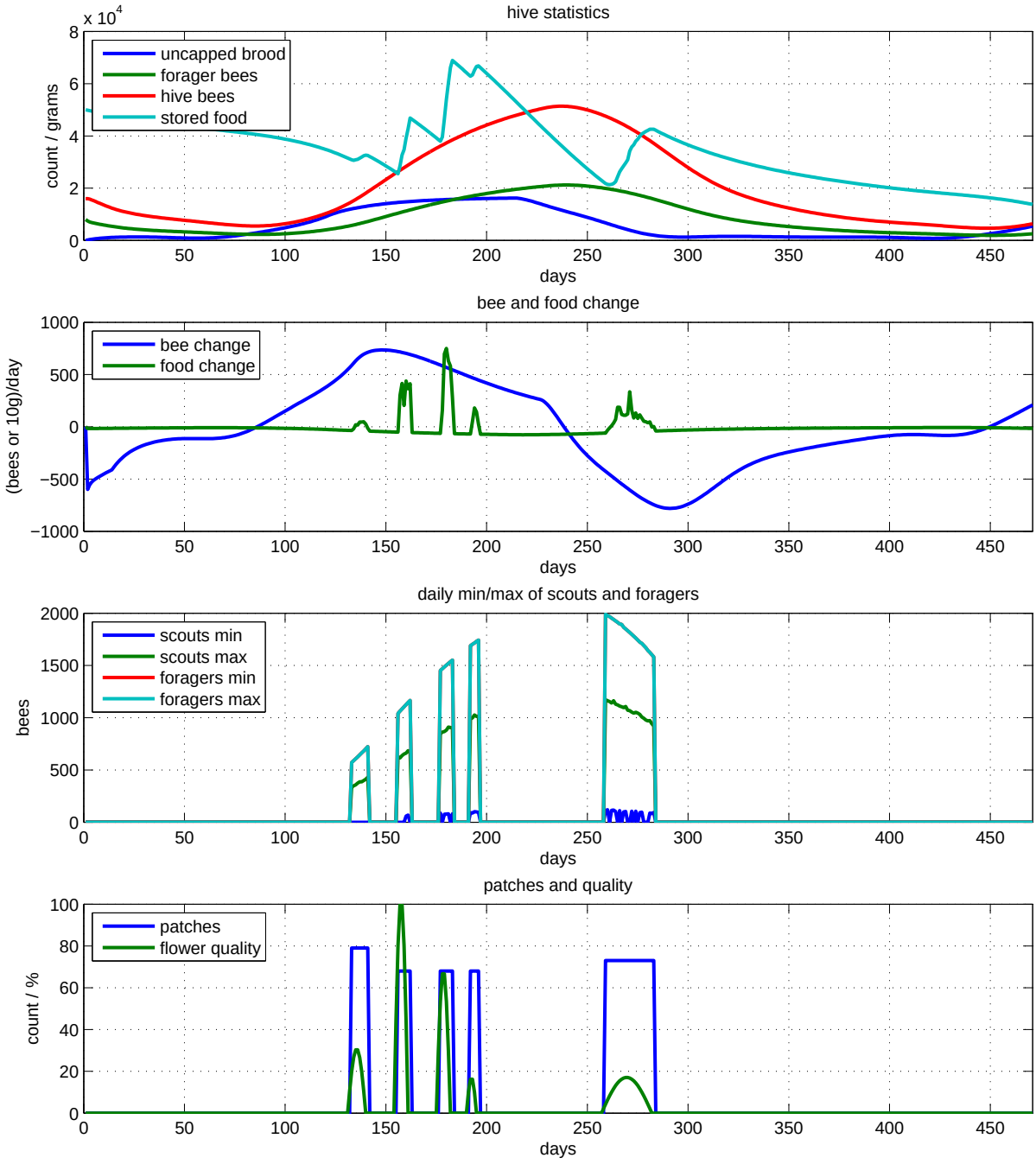
A.3.7 Peak = 1.5, delayed by 20 days



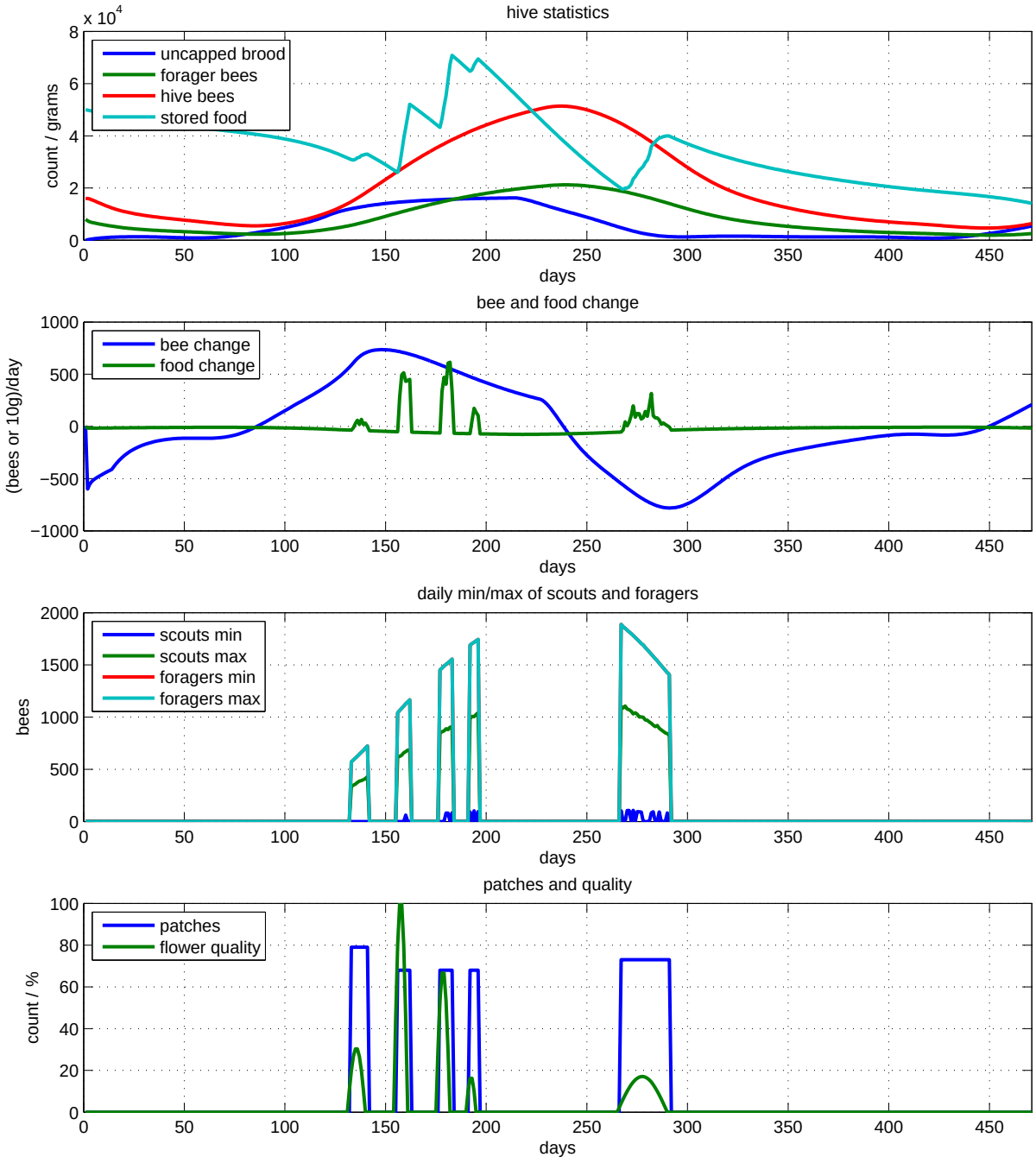
A.3.8 Peak = 1, delayed by 4 days



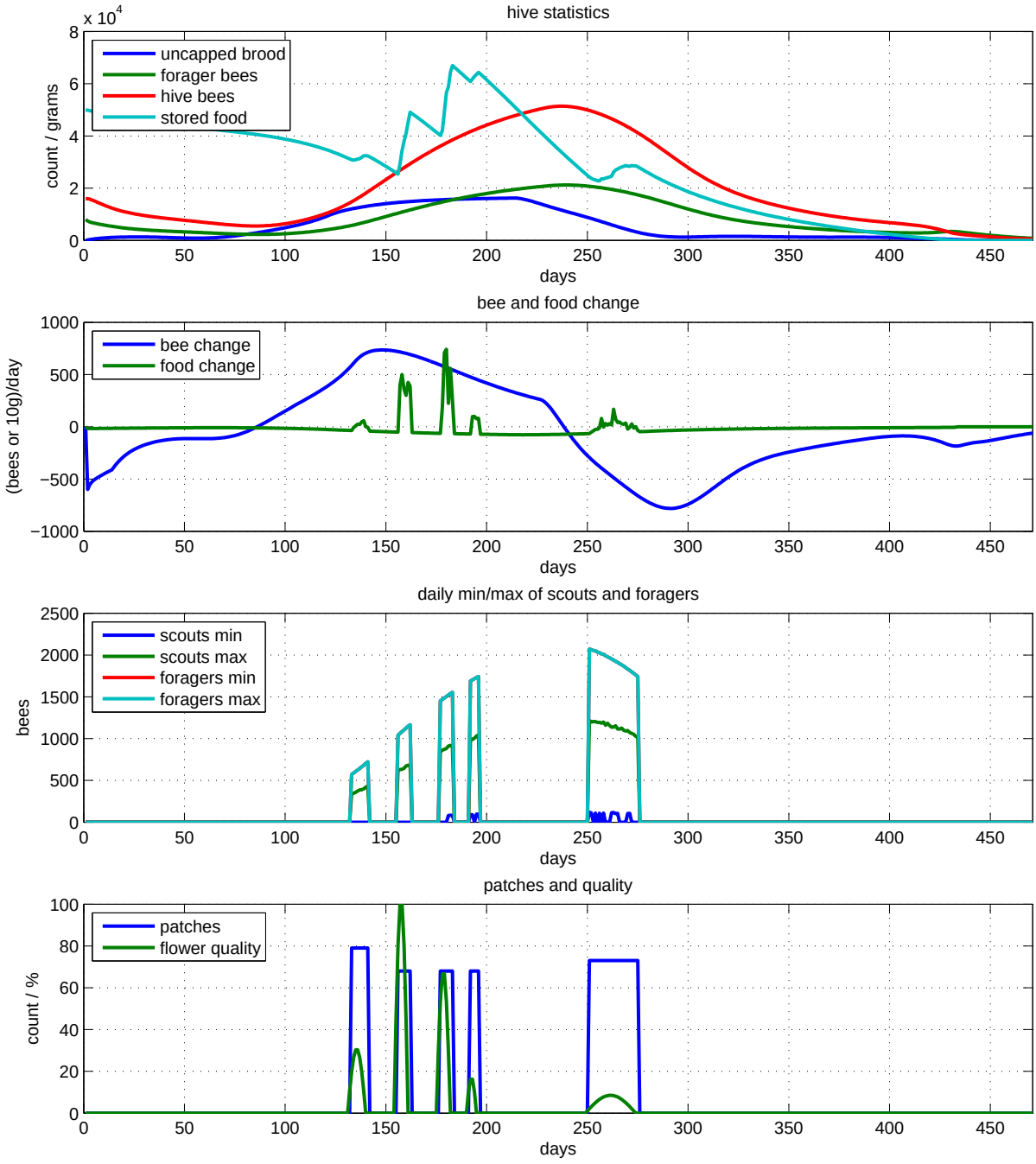
A.3.9 Peak = 1, delayed by 12 days



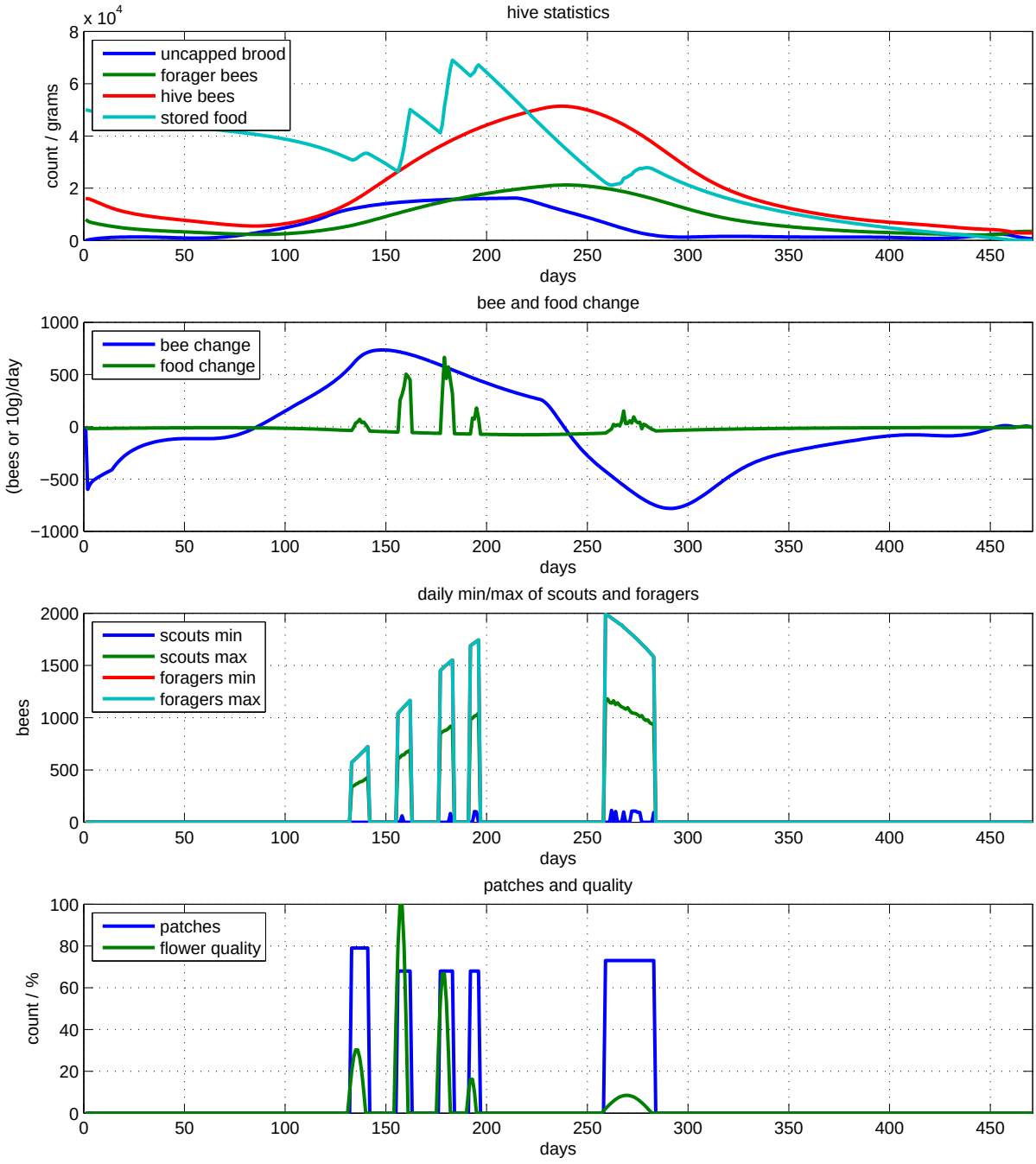
A.3.10 Peak = 1, delayed by 20 days



A.3.11 Peak = 0.5, delayed by 4 days



A.3.12 Peak = 0.5, delayed by 12 days



A.3.13 Peak = 0.5, delayed by 20 days