"HAPPY WIFE, HAPPY LIFE": MODELING HAPPINESS IN RELATIONSHIPS

P. BEAL, T. EDDINGTON, T. HART, M. VINES, A. WARD

Abstract. In this paper, we explore methods of modeling emotional interaction between individuals using differential equations. We examine expectations about the influence of one's emotional state on those around them through psychological and social science research and then build models to show the implications and long-term system dynamics that result from the conditions shown in research. We also generalize the model to handle an arbitrary group of people and model the dynamics between them all. Our findings include a description of the combinations of initial conditions and descriptions that predict long-term happiness and growth in a relationship.

1. Background/Motivation

From our personal experiences, we have observed that a person's happiness is affected by his or her relationships with other people (e.g. siblings, parents, a partner, etc.) [AMHea13, Ken12]. This observation prompts the question, how can the happiness of two people who are in a relationship be modeled using ordinary differential equations? This is the foundation of our project–to use differential equations to model the way that relationships and interactions affect an individual's happiness over time.

Our motivation for the following model and experiments stems from three main questions:

(1) How does the way that two people interact in a relationship affect or predict their long-term happiness?
(2) Under what conditions does the adage "happy wife, happy life" actually hold? That is, which parameters will lead to long-term increasing happiness in a relationship?
(3) Is there a sort of "universal strategy" for a happy relationship? In mathematical terms, can we identify a set of parameters that produce a favorable stable solution regardless of initial conditions? Further, can we identify initial conditions that produce a favorable stable solution regardless of the system parameters?

We will experiment using a variety of parameters to adjust our model to answer different questions, as well as expand the model to a general case with more than two people.

Date: December 7, 2024.

2. Modeling

2.1. **Derivation.** We begin the derivation of our model with a few basic assumptions:

(1) An individual's happiness is quantifiable and time-dependent [Gra12].
(2) The fluctuation of an individual's happiness in the absence of relationships is known and can be modeled by some function of time [Bro21].
(3) In general, the happiness levels of two people in a relationship will change in proportion to the difference between their happiness. That is, one gets happier as they interact with an individual who is more happy, and less happy when interacting with someone who is less happy [Cen24, BC11, SR15].

Beginning in the 2-person model, we represent the happiness of Person 1 and Person 2, respectively, by $h_1(t)$ and $h_2(t)$. Thus, we seek to model the derivatives $\dot{h}_1(t)$ and $\dot{h}_2(t)$. Because we assume a person's individual (in the absence of relationships) change in happiness is known, we say that each $\dot{h}_i(t)$ consists of two components: personal happiness fluctuation $\dot{p}_i(t)$, and relationship happiness fluctuation $\dot{r}_i(t)$. With these two components, we give the following basic form:

$$\dot{h}_i(t) = \dot{p}_i(t) + \dot{r}_i(t).$$

The component $\dot{p}_i(t)$ is assumed, and in this project takes two general forms: we can assume a person's individual happiness level to be either constant or periodic [Bro21, SK20]. Of course, there are many other functions that could model individual happiness, but we limit the scope of this paper to these two cases. We therefore define $p_i(t) = -a\sin(ct)$, where a defines the amplitude and c defines the period of fluctuation, so an amplitude of $a = 0$ (or a period of $c = 0$) corresponds to constant p_i. This definition gives $\dot{p}_i(t) = -ac\cos(ct)$. We may also add another periodic term $-b\sin(dt)$ to $p_i(t)$, describing a small scale fluctuation of happiness that changes frequently (see 3.3), such as due to emotional disorders [Gol11]. This definition gives $\dot{p}_i(t) = -ac\cos(ct) - bd\cos(dt)$.

Because we expect two people's happiness levels to tend towards each other proportionally to the difference between their happiness levels, we set

$$\dot{r}_1 = \gamma_1(h_2 - h_1)$$
$$\dot{r}_2 = \gamma_2(h_1 - h_2)$$

where the coefficient γ_i describes the reactivity (how much one person's happiness is subject to change based on the other's happiness) of a person to his or her counterpart's happiness. A gamma value of 0 represents a person whose happiness is uninfluenced by his or her interactions, while a gamma value closer to 1 or -1 indicates a high level of positive or negative reactivity. For ease of analysis we want to bound our solutions, so we will multiply each $\dot{h}_i$ by factors of $\left(1 - \frac{h_i}{k}\right)\left(1 + \frac{h_i}{k}\right) = 1 - \frac{h_i^2}{k^2}$. This constrains

the solutions to the range $[-k, k]$, provided the initial conditions also lie within this range. For our model, we choose $k = 10$ (see 2.2).
Using these components, we give the following general model for the 2-person scenario:

$$\dot{h}_1(t) = (-ac\cos(ct) - bd\cos(dt) + \gamma_1(h_2 - h_1))\left(1 - \frac{h_i^2}{k^2}\right)$$
$$\dot{h}_2(t) = (-ac\cos(ct) - bd\cos(dt) + \gamma_2(h_1 - h_2))\left(1 - \frac{h_i^2}{k^2}\right)$$

Note: The amplitude and frequency parameters a, b, c, and d need not be the same between Person 1 and Person 2. However, in order to avoid using too many subscripts, we notate them with the same letter here.

Our ODE model contains many parameters, summarized below.

- $\boldsymbol{a}, \boldsymbol{c}$ – The amplitude and period, respectively, of individual happiness fluctuation with $a \in [0, k]$, $c \in [0, \infty)$.
- $\boldsymbol{\gamma}$ – The reactivity coefficient, with $\gamma \in [-1, 1]$.
- $\boldsymbol{k}$ – The "carrying capacity" for individual happiness, bounding the solutions to $[-k, k]$.
- $\boldsymbol{t_0}$ – The time at which a relationship between Person 1 and Person 2 begins. If $t_0 > 0$, the components $\dot{r}_i(t)$ will be multiplied by the indicator function $\mathbb{1}_{[t_0,\infty)}(t)$.
- $\boldsymbol{t_f}$ – Represents how long it takes for two people to adjust to their relationship and become dependent. If $t_f > 0$, the component $\dot{r}_i(t)$ is scaled by $\left(1 - \frac{t_f - t}{t_f - t_0}\right)$. This allows the relational component of happiness to take effect gradually.
- $\boldsymbol{b}, \boldsymbol{d}$ – The amplitude and period, respectively, for small scale happiness fluctuations, with $b \in [0, k]$, $d \in [0, \infty)$ (see 3.3).

2.2. **Unit Analysis.** For our model, we are measuring "Happiness Units", which we choose to interpret as if someone were to take a survey and rate their personal happiness on a scale from -10 to 10, with -10 being extremely unhappy and 10 being extremely happy. The "Happiness Units" will be denoted H, so $[h_i(t)] = \mathsf{H}$ and $[\dot{h}_i(t)] = \mathsf{HT}^{-1}$. This means $[\dot{p}_i(t)] = \mathsf{HT}^{-1}$ and $[\dot{r}_i(t)] = \mathsf{HT}^{-1}$ since $\dot{h}_i(t) = \dot{p}_i(t) + \dot{r}_i(t)$.

Now, $p_i(t) = -a\sin(ct) - b\sin(dt)$. We let $[a] = [b] = \mathsf{H}$ and $[c] = [d] = \mathsf{T}^{-1}$ so that $[p_i(t)] = \mathsf{H}$. We choose this because a and b represent the happiness amplitude and c and d are related to a frequency. Also, $\dot{r}_1 = \gamma_1(h_2 - h_1)$, which implies $[\gamma] = \mathsf{T}^{-1}$ so that $[\dot{r}_i(t)] = \mathsf{HT}^{-1}$.

Finally, since k represents a level of happiness it has units $[k] = \mathsf{H}$. This means that in the model, $\frac{h_i^2}{k^2}$ is unitless, which makes $\left(1 - \frac{h_i^2}{k^2}\right)$ commensurable in the standard model above.

2.3. **Generalized Case.** We now examine a general case with more than two people. To simplify the analysis, we focus on the case that $p_i(t) = k_i$

where each $k_i \in \mathbb{R}$ (i.e. we represent individual happiness as constant in time). Therefore, $\dot{p}_i(t) = 0$. We also make the assumption that $\dot{r}_i(t)$ depends on all h_j. Using the assumptions described earlier, this gives the general model

$$\dot{h}_i = \left(1 - h_i^2\right) \sum_{j=1}^{n} \gamma_{ji}(h_j - h_i) = f(\mathbf{h})_i, \tag{1}$$

where γ_{ji} is the reactivity coefficient between person j and person i, or the degree to which person j's emotions influence person i. Also, note that when $i = j$, the term $h_j - h_i$ vanishes, so the γ_{ii} is irrelevant. Because it does not affect the dynamics of the system, we will assume that $\gamma_{ii} = 0$ for all i.

3. Results

3.1. **Constant Gamma Experiment.** There are some results we would expect to see as the reactivity parameter varies [Cen24, AMHea13]. First, we would expect that if someone has a positive reactivity, their happiness trends toward the other person's happiness. Likewise, if a person has a negative reactivity, their happiness trends away from the other person's happiness. We now look at a few simple cases.

3.1.1. *Case 0:* $\gamma_1 = \gamma_2 = 0$. In this trivial case, neither person's level of happiness affects the other person, so they each remain constant in their level of happiness (no plot).

3.1.2. *Case 1:* $\gamma_1, \gamma_2 < 0$. Each person is negatively impacted by the difference between the other person's happiness and their own, leading to the levels of happiness moving apart and eventually settling at opposite ends of the possible spectrum, so whoever has the higher starting condition will converge to the maximum level of happiness while the other person will converge to the minimum.

3.1.3. *Case 2:* $\gamma_1 < 0, \gamma_2 > 0$. This case models an interaction between one anxiously attached person and one avoidant person. The avoidant person in this case seeks to move apart from the happiness level of the other person, while the anxious person seeks to move closer to their partner. Ultimately, this leads to both people together converging to one boundary, depending on the initial conditions. If the avoidant starts off happier, they both become happier, while if the avoidant starts off less happy, they both become less sad, converging to the minimum.

3.1.4. *Case 3:* $\gamma_1, \gamma_2 > 0$. Here, the people move towards the happiness level of the other person, eventually converging to some point between their initial level of happiness.

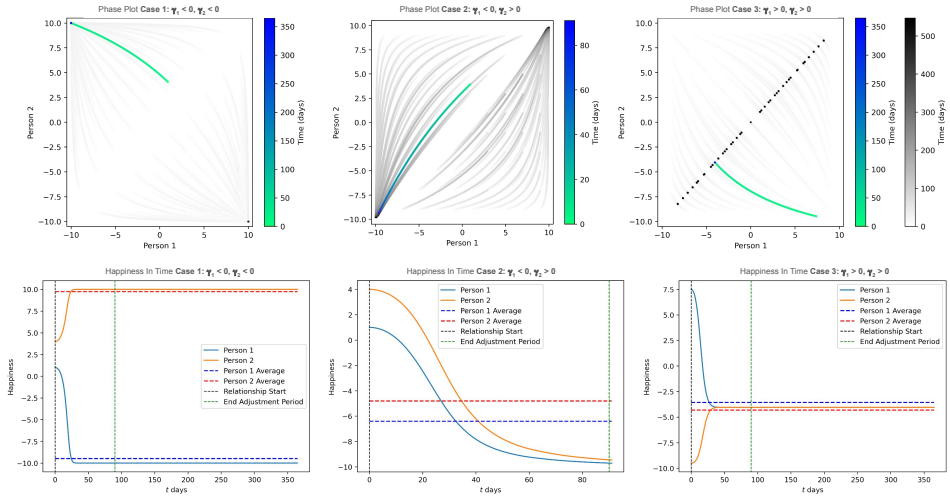


FIGURE 1. Phase plots and time-series plots for the interaction between two people with varying signs of the γ parameters. The blue-green line in the phase plots corresponds to the initial conditions shown in the corresponding time-series plots.

3.2. **Non-Constant Gamma Experiment.** Considering that people may be more or less reactive in a relationship based on different factors [AMHea13], we explore implementing the reactivity coefficient γ as a function of time, representing varying types of reactivity in people.
One type of reactivity will be a person who increases reactivity until they hit a threshold. We model this with a logistic equation. One situation of this could be someone who depends more on their partner as they grow closer or have known them longer.

$$\gamma(t) = \frac{L}{1 + e^{-k(t-\zeta)}} - s \tag{2}$$

Another person may have oscillating reactivity over time, and we represent this with the sine equation. For plots for experiments on this function look at the code.

$$\gamma(t) = \alpha \sin(\nu t) + \beta \tag{3}$$

The last variation of a non-constant gamma we will explore is reactivity that increases and the decreases back to the original starting value, we show this by the normal distribution represented by $f_x(x)$. This is a person who gets more reactive based on their partner for a certain time and then eases back to their original reactivity level, possibly due to comfort in the relationship or some other factor. For plots for experiments on this function look at the

code.

$$\gamma(t) = \rho f_x(t) + \eta \tag{4}$$

For simplicity we assume: time period of 2 years, starting happiness values are (0.5, 0.8), happiness amplitude is (0.2, 0.3), and day the two people interact is the 100th. We model the situations where each type of non-constant reaction is paired with someone of constant reactivity, the same type of reactivity, and the negative version of their reactivity. For best comparison, we will show these situations modeled with equation (2).

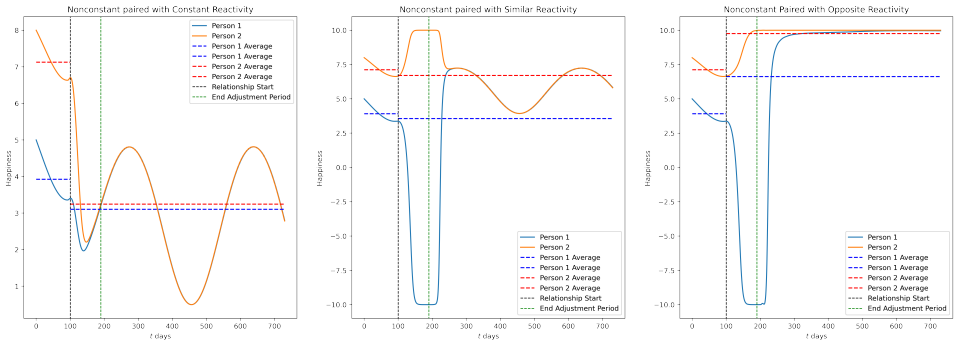


FIGURE 2. We fix $t_0 = 100$, and give logistic parameters $L = 0.5$, $k = 0.3$, $\zeta = 200$ days (so full reactivity occurs 100 days after the relationship starts), and $s = 0.2$. This reactivity function lies between -0.2 and 0.3

(1) Logistic Reactivity Paired with Constant Reactivity: As you can see by Figure 2i), Person 1 initially trends away from Person 2 because they start with negative reactivity and Person 2 continues to chase them once their relationship starts. As Person 1's reactivity becomes larger, their change in happiness converges towards each other.
(2) Logistic Reactivity Paired with the Same Reactivity: Because both people start with negative reactivity their happiness initially grows apart. As their reactivity grows, their change in happiness converges towards each other.
(3) Logistic Reactivity Paired with Negative Reactivity: As Figure 2iii) shows, Person 2 doesn't trend towards Person 1 even though they have positive reactivity at first, but they do increase slowly and stay at maximum happiness. This is because as Person 1's reactivity becomes larger they grow toward Person 2. Person 2 wants to grow apart from their partner since they settle with negative reactivity, but they have already reached capacity.

3.3. **Abnormal Fluctuation Experiments.** As found in [Gol11, Bro21], individuals face the ups and downs of day-to-day life which affect happiness on a relatively short term. This type of rapid change could also be influenced

by emotional disorders such as depression, anxiety, bipolar disorder, or a host of other personal challenges that many individuals face.

In light of these realities, we use the parameters b and d to model the amplitude and period, respectively, of small scale fluctuations in happiness. In these experiments, we fix all other parameters and will only consider changes in b and d. Let $a_1 = 0.3$, $a_2 = 0.6$, $c_1 = c_2 = \frac{2\pi}{365}$, $\gamma_1 = .01$, $\gamma_2 = .05$, $t_0 = 547$ (approximately 1.5 years), and $t_f = 182$ (approximately 0.5 years). We will choose base values of b and d for both individuals, given by $b_1 = 0.05$, $b_2 = 0.2$, $d_1 = \frac{2\pi}{30}$, and $d_2 = \frac{2\pi}{90}$. We then change the amplitude and period of each person's fluctuation separately to observe the impact of each change. The solution with these base values is shown in Figure 3.

In other words, we have this general case: Person 1's happiness cycles on a yearly basis with a small amplitude. They have relatively low reactivity, and their happiness fluctuates locally approximately every month (30 days) with small amplitude. On the other hand, Person 2's happiness cycles on a yearly basis with double the amplitude of Person 1. They have a relatively high reactivity, and their individual happiness fluctuates locally approximately every 3 months (90 days) with a higher amplitude.

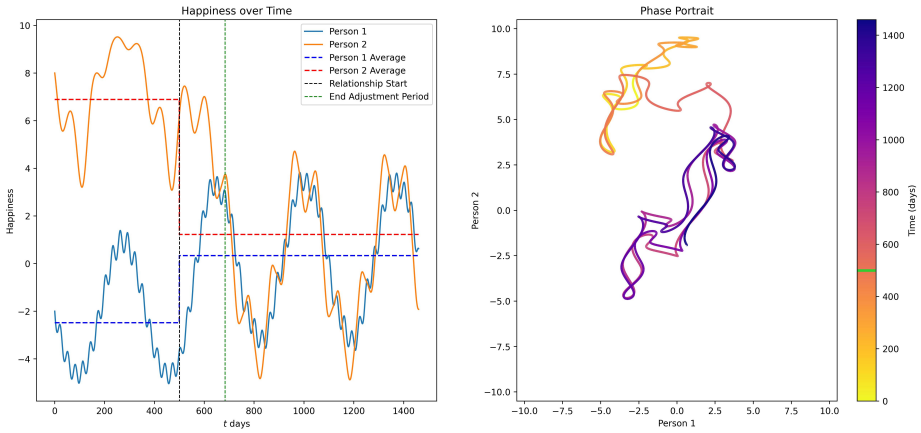


FIGURE 3. The base model using non-zero b and d parameters with which we will experiment.

3.3.1. *Expected Behavior.* Based on our model parameters, we describe the expected model behavior when we increase each parameter within its possible values.

$\boldsymbol{b_1}$ – As b_1 increases, we expect that Person 2's happiness will likewise experience an amplitude increase. This is because Person 2 has a relatively high reactivity coefficient, and will thus more closely follow the happiness of Person 1.

$\boldsymbol{b_2}$ – As b_2 increases, we expect that Person 1's happiness will experience a small increase in amplitude. However, because of the relatively low reactivity

coefficient, Person 1's happiness will not track as closely with the happiness of Person 2.

$\boldsymbol{d_1}$ – Since Person 2 has a relatively large reactivity coefficient, as d_1 increases, we expect that Person 2's happiness will exhibit more stable behavior as d_1 approaches $c_1 = \frac{2\pi}{365}$ since it will be drawn closer to the other person's stabilizing happiness level.

$\boldsymbol{d_2}$ – Since Person 1 has a relatively small reactivity coefficient, as we increase the period of Person 2's small scale fluctuations, we expect that Person 1's happiness will not change greatly as d_2 approaches $c_2 = \frac{2\pi}{365}$.

3.3.2. *Observed Behavior.* We obtained observations for each of these cases by animating the solutions over each range of values. We found the following:

$\boldsymbol{b_1}$ – Contrary to our expectation, as b_1 increased, the amplitude for Person 2 did not increase. Instead, the change in d_1 led to a change in the period of small scale fluctuations for Person 2! Since Person 2 has a higher reactivity coefficient, we observed new peaks and valleys in Person 2's happiness in between the peaks and valleys already present in the base model.

$\boldsymbol{b_2}$ – As b_2 increased, we observed what we expected. The solution stretched slightly towards the peaks and valleys, but the amplitude and period of the small scale fluctuations essentially remained the same.

$\boldsymbol{d_1}$ – Since Person 1 has a smaller reactivity coefficient, as d_1 approached c_1, the yearly fluctuations swamped the behavior of the solution, and the small scale fluctuations largely disappeared, even in the context of Person 2's small scale fluctuations. Contrary to our expectation, Person 2 saw largely no change in amplitude or period of the small scale fluctuations throughout the experiment.

$\boldsymbol{d_2}$ – As we expected, the happiness levels of Person 1 stayed relatively the same as d_2 increased. However, unexpectedly as d_2 approached c_2, Person 2's happiness began to be more impacted by the small scale fluctuations of Person 1, due to the higher reactivity coefficient of Person 2. This led to quasi-small scale fluctuations appearing in Person 2's happiness, even when the long term fluctuations had already swamped the overall happiness of Person 2.

3.4. **Generalized Model.** When we include many people in the model, the results generalize as expected. We do not have sufficient time to cover the many possibilities of varying γ parameters or individual changes in happiness in addition to the influence of other people, so we share one specific example of a time-series plot (see Figure 4) showing the happiness interactions between four people, using γ values of

$$\boldsymbol{\gamma} = \frac{1}{100}\begin{bmatrix} 0 & 0.1 & -0.1 & -0.1 \\ 0 & 0 & 0.05 & 0.5 \\ 0 & 0.2 & 0 & -0.2 \\ -0.5 & -0.2 & 0 & 0 \end{bmatrix},$$

where $[\boldsymbol{\gamma}]_{ji} = \gamma_{ji}$ is the reactionary coefficient between person j and person i, or the degree to which person j's happiness affects person i. Note in the plot that because of the more complex interactions between people, the levels of happiness can now cross paths.

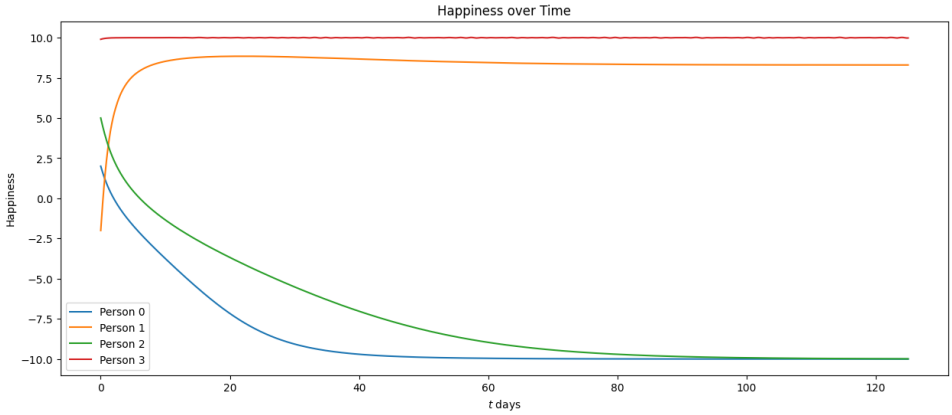


FIGURE 4. Generalized model of happiness interaction for four people. The initial conditions were $h_1 = 0.2, h_2 = -0.2, h_3 = 0.5$, and $h_4 = 0.99$.

4. Analysis/Conclusions

4.1. **Research Question Analysis.** While we acknowledge that this model does not fit reality perfectly, it provides behavior that simultaneously matches our expectations about happiness in a relationship and gives unexpected insights into our research questions. In relation to our first question, we found that the main influence on long-term happiness came from the individuals' attachment styles, modeled by the reactivity coefficient γ. The final state of happiness for each individual trended towards the results found in Section 3.1.3 for the different combinations of positive and negative γ, even when a varying γ value or abnormal happiness fluctuations were taken into account as shown in Figures 2 and 3.

Secondly, our modeling shows that the adage "happy wife, happy life" holds in specific circumstances, namely, if the woman in the relationship starts with a naturally higher level of individual happiness, and the man has a positive reactivity coefficient γ. This is because we found that happiness tended to average out between the individuals, so the person who started lower was brought higher. Thus, to also answer the third question, the recommended strategy from our modeling for a happy, stable relationship would be to start with the same level of happiness as your partner and be positively reactive towards each other. This means that each person's happiness would not be affected at all by the relationship and would remain on it's current trajectory. Another strategy is for one person to have a higher

starting happiness and negative γ and the other to have a lower starting happiness and positive γ, as mentioned in 3.1.3. However, we recognize that implementing these strategies is not very realistic, which is a key limitation of our modeling process.

4.2. **Future Explorations.** One of the main unrealistic results obtained from our model is that in almost all cases, the person who starts with the higher level of happiness is brought lower when in a relationship. This led to unrealistic answers to Questions 2 and 3 that contradict our intuition that when two generally happy people are in a relationship, both of their happiness levels should increase. In the future, the simplifying assumptions made in our modeling could be changed to strive to incorporate these ideas. Also, a different question that we originally considered was: How does happiness in a relationship change through major life events? This would be an interesting research topic since there are various life events such as marriage, a death in the family, etc. that could impact the happiness within a relationship. In future research we would also like to include this capability into the model.

4.3. **Modeling Challenges.** The model faces some numerical stability challenges. For instance, certain small choices of d_i lead to dramatic floating point error. If $d_2 = \frac{2\pi}{7}$ and $d_1 = \frac{2\pi}{25}$, the solutions blow up around $t = 715$, as shown in Figure 5. This phenomenon also occurs when $d_1 = \frac{2\pi}{91}$ or $\frac{2\pi}{177}$, among others. This problem is mitigated when d_2 is raised to $\frac{2\pi}{30}$ (hence why it was chosen as the base value for d_2).

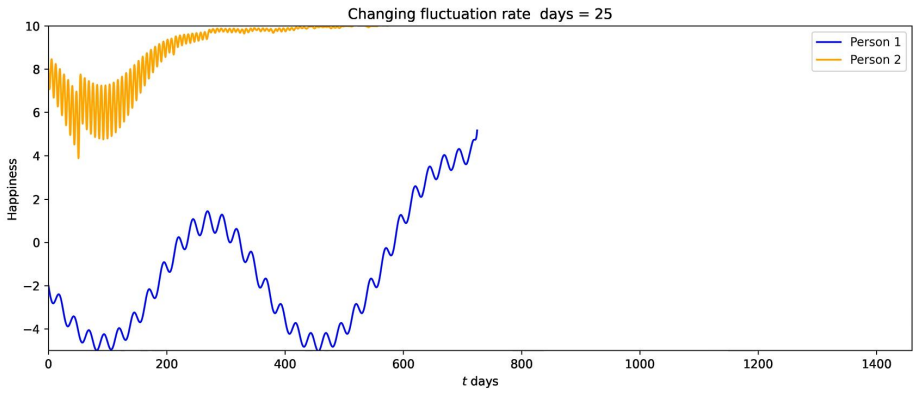


FIGURE 5. Example of numerical instability with $d_1 = \frac{2\pi}{25}$ and $d_2 = \frac{2\pi}{7}$. Person 2 experiences a floating-point error change in amplitude around $t = 300$, which then causes the solution to break the boundaries of the model. The solution never recovers and disappears to infinity.

References

[AMHea13] Carolyn Korber Angela M. Hicks et al. Attachment theory. *Encyclopedia of Behavioral Medicine*, pages 149–155, 2013.

[BC11] Lane Beckes and James A. Coan. Social baseline theory: The role of social proximity in emotion and economy of action. *Social and Personality Psychology Compass*, 5:976–988, 12 2011.

[Bro21] Tracy Brower. Happiness ebbs and flows, 4 2021.

[Cen24] Kaizen Center. How do relationships impact our happiness?, 2 2024.

[Gol11] Albert Goldbeter. A model for the dynamics of bipolar disorders. *Progress in Biophysics and Molecular Biology*, 3 2011.

[Gra12] Carol Graham. How can we effectively measure happiness?, 10 2012.

[Ken12] Stacey Kennelly. How to be happy: The fine print, 8 2012.

[SK20] Dan Pilat Sekoul Krastev. Hedonic treadmill, 8 2020.

[SR15] Dominik Schoebi and Ashley K. Randall. Emotional dynamics in intimate relationships. *http://dx.doi.org/10.1177/1754073915590620*, 7:342–348, 7 2015.

In [1]:
```
import numpy as np
import matplotlib.pyplot as plt
from scipy.integrate import solve_ivp
import warnings
from matplotlib.collections import LineCollection
plt.rcParams['figure.figsize'] = (18, 8)
plt.rcParams['figure.dpi'] = 350
from scipy.stats import norm
```

In [2]:
```
def colored_line(x, y, c, ax, **lc_kwargs):
    """
    Plot a line with a color specified along the line by a third value.

    It does this by creating a collection of line segments. Each line segment is
    made up of two straight lines each connecting the current (x, y) point to the
    midpoints of the lines connecting the current point with its two neighbors.
    This creates a smooth line with no gaps between the line segments.

    Parameters
    ----------
    x, y : array-like
        The horizontal and vertical coordinates of the data points.
    c : array-like
        The color values, which should be the same size as x and y.
    ax : Axes
        Axis object on which to plot the colored line.
    **lc_kwargs
        Any additional arguments to pass to matplotlib.collections.LineCollection
        constructor. This should not include the array keyword argument because
        that is set to the color argument. If provided, it will be overridden.

    Returns
    -------
    matplotlib.collections.LineCollection
        The generated line collection representing the colored line.
    """
    if "array" in lc_kwargs:
        warnings.warn('The provided "array" keyword argument will be overridden')

    # Default the capstyle to butt so that the line segments smoothly line up
    default_kwargs = {"capstyle": "butt"}
    default_kwargs.update(lc_kwargs)

    # Compute the midpoints of the line segments. Include the first and last points
    # twice so we don't need any special syntax later to handle them.
    x = np.asarray(x)
    y = np.asarray(y)
    x_midpts = np.hstack((x[0], 0.5 * (x[1:] + x[:-1]), x[-1]))
    y_midpts = np.hstack((y[0], 0.5 * (y[1:] + y[:-1]), y[-1]))

    # Determine the start, middle, and end coordinate pair of each line segment.
    # Use the reshape to add an extra dimension so each pair of points is in its
    # own list. Then concatenate them to create:
    # [
```

```
    #   [(x1_start, y1_start), (x1_mid, y1_mid), (x1_end, y1_end)],
    #   [(x2_start, y2_start), (x2_mid, y2_mid), (x2_end, y2_end)],
    #   ...
    # ]
    coord_start = np.column_stack((x_midpts[:-1], y_midpts[:-1]))[:, np.newaxis, :]
    coord_mid = np.column_stack((x, y))[:, np.newaxis, :]
    coord_end = np.column_stack((x_midpts[1:], y_midpts[1:]))[:, np.newaxis, :]
    segments = np.concatenate((coord_start, coord_mid, coord_end), axis=1)

    lc = LineCollection(segments, **default_kwargs)
    lc.set_array(c)  # set the colors of each segment

    return ax.add_collection(lc)
```

In [28]:

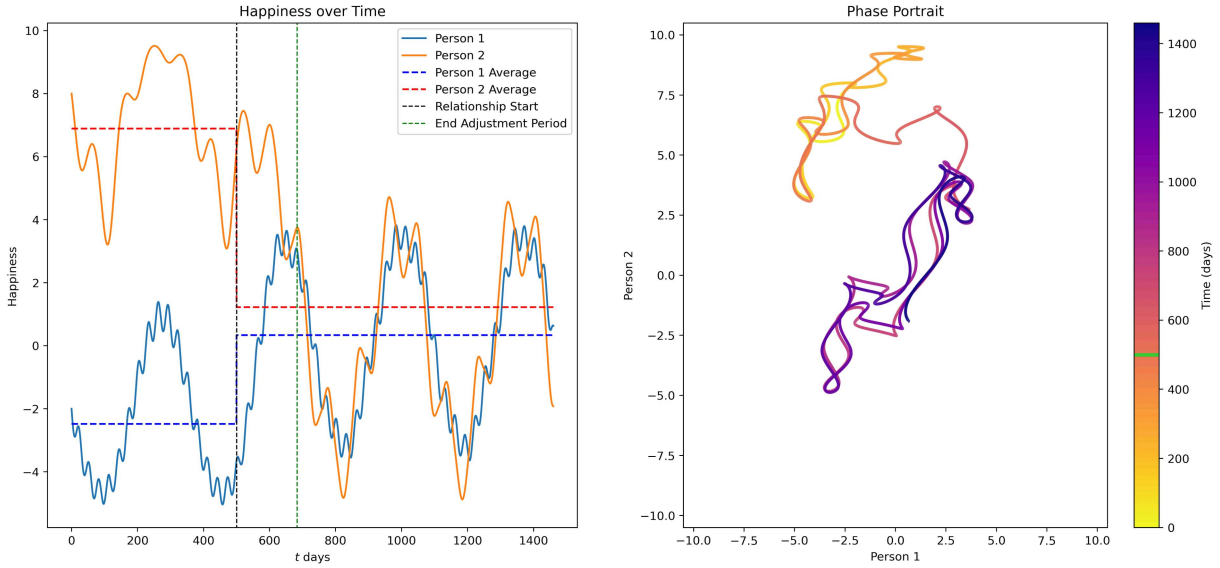
```
def base_model(x0=np.array([0.5, 0.5]), gamma=np.array([0.5, 0.5]), c=np.array([2*n
               d=np.array([2*np.pi / 7, 2*np.pi / 7]), amp=np.array([0, 0]), week_a
               t0=0, days_adjust=90, num_years=1, plot=True, plot_avg_lines=True, p
    '''Return the approximated solution to the two-person relationship model under
       1. The personal happiness of each individual is either constant or periodic
       2. The level of reactivity of each individual is measured by a constant.

       Parameters:
           x0 ((1,2) ndarray) : the initial conditions, i.e. starting happiness va
               - defaults to 0.5 for both people
           gamma ((1,2) array-like) : the reactivity coefficients for persons 1 an
               - defaults to 0.5 for both people
           c ((1, 2) array-like) : the period of individual happiness fluctuation
               - defaults to 2*np.pi / 365 (happiness fluctuates seasonally)
           amp ((1, 2) array-like) : dictates the amplitude of each person's indiv
               - defaults to (0, 0) (corresponding to constant invidiual happiness
           d ((1, 2) array-like) : the period of individual happiness fluctuation
                       on a small-scale basis
               - defaults to 2*np.pi / 365 (happiness fluctuates seasonally)
           week_amp ((1, 2) array-like) : dictates the amplitude of each person's
                       wave on a small-scale basis
               - defaults to (0, 0) (corresponding to constant invidiual happiness
           t0 (float) : the time at which persons 1 and 2 begin interacting
               - defaults to 0 (always in relationship)
           days_adjust (int) : the number of days for both people to adjust to rel
                       i.e. reach full interactivity
               - defaults to 90
           num_years (float) : the length of the time period to evaluate, in years
               - defaults to 1
           plot (bool) : whether to show solution plots in addition to returning t
               - defaults to True
           plot_avg_lines (bool) : whether to show the average happiness lines on
               - defaults to True
           plot_event_lines (bool) : whether to plot the vertical lines indicating
                       the relationship and end of adjustment phase
               - defaults to True

       Returns:
           (ndarray, ndarray, ndarray) : the approximated solution to the IVP
               - includes time steps and approximated values for person 1 and pers
    '''
    # generate the time space
```

```
    T = int(num_years * 365)
    t_span = (0, T)
    t_vals = np.linspace(0, T, 3*T + 1)

    # the indicator function determines whether the individuals are interacting
    ind = lambda t : int(t >= t0)
    tf = t0 + days_adjust

    # function that implements an adjustment period after the start of the relation
    def _adjust(t):
        if t < t0: return 0
        elif t > tf: return 1
        else: return 1 - (tf - t) / (tf - t0)

    # define the ODE
    def base_ode(t, x):
        # personal happiness components
        H1p_dot= -1 * (amp[0] * c[0] * np.cos(c[0] * t)
                       + week_amp[0] * d[0] * np.cos(d[0] * t))
        H2p_dot = -1 * (amp[1] * c[1] * np.cos(c[1] * t)
                        + week_amp[1] * d[1] * np.cos(d[1] * t))

        # Total happiness
        H1t_dot = H1p_dot + gamma[0] * (x[1] - x[0]) * ind(t) * _adjust(t)
        H2t_dot = H2p_dot + gamma[1] * (x[0] - x[1]) * ind(t) * _adjust(t)

        # return -- scale by 'carrying capacity' to keep solutions bounded
        return np.array([
            H1t_dot * (1 - x[0] / 1) * (1 + x[0] / 1),
            H2t_dot * (1 - x[1] / 1) * (1 + x[1] / 1)
        ])

    # approximate solution to IVP
    sol = solve_ivp(base_ode, t_span, x0, t_eval=t_vals)
    p1_vals = sol.y[0] * 10
    p2_vals = sol.y[1] * 10

    # plot solutions if desired
    if plot:
        # scale happiness levels to 0 - 10, like survey results
        fig1, axs = plt.subplots(nrows=1, ncols=2)
        axs[0] = plt.subplot(121)
        axs[0].plot(sol.t, p1_vals, label='Person 1')
        axs[0].plot(sol.t, p2_vals, label='Person 2')

        # plot the average lines and relationship event lines
        index = int(t0 * (3*T + 1) / T)
        if plot_avg_lines:
            axs[0].plot(sol.t,
                        [np.mean(p1_vals[:index])] * index + [np.mean(p1_vals[index
                        'b--',
                        label="Person 1 Average")
            axs[0].plot(sol.t,
                        [np.mean(p2_vals[:index])] * index + [np.mean(p2_vals[index
                        'r--',
                        label="Person 2 Average")
```

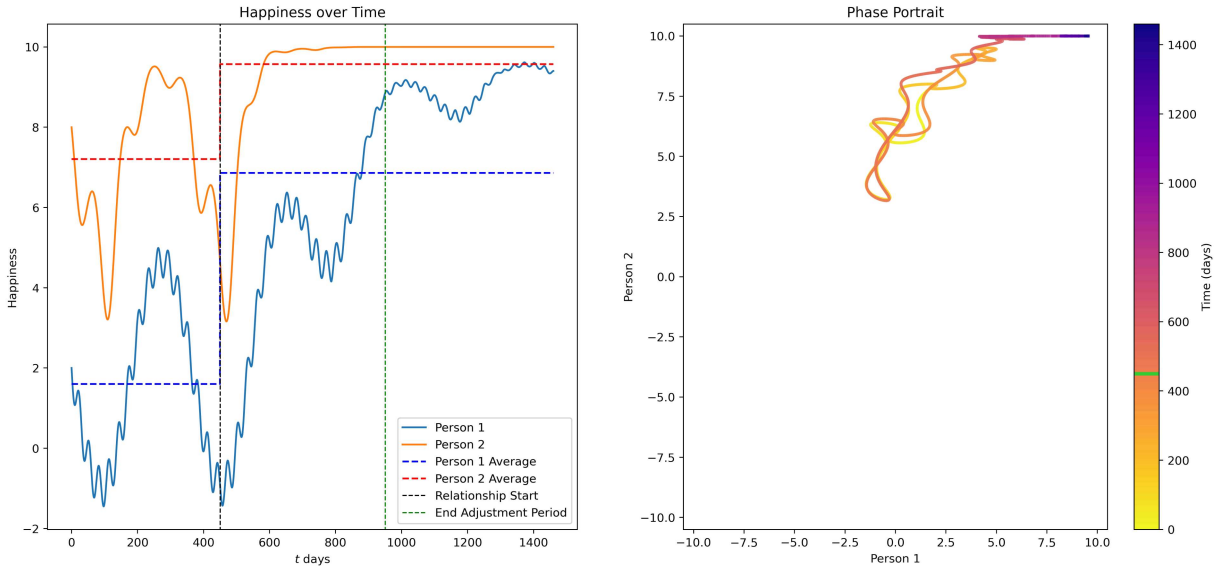
```
        if plot_event_lines:
            axs[0].axvline(t0,
                           ymin=min(min(p1_vals), min(p2_vals)) - .5,
                           ymax=max(max(p1_vals), max(p2_vals)) + .5,
                           c='k',
                           ls='--',
                           lw=1,
                           label="Relationship Start")
            axs[0].axvline(tf,
                           ymin=min(min(p1_vals), min(p2_vals)) - .5,
                           ymax=max(max(p1_vals), max(p2_vals)) + .5,
                           c='g',
                           ls='--',
                           lw=1,
                           label="End Adjustment Period")

        # finish the happiness plot
        axs[0].set_xlabel(r"$t$ days")
        axs[0].set_ylabel("Happiness")
        axs[0].legend()
        axs[0].set_title("Happiness over Time")

        # plot the phase portrait
        lines = colored_line(sol.y[0]*10,
                             sol.y[1]*10,
                             np.linspace(0, sol.t[-1], len(sol.t)+1),
                             axs[1],
                             linewidth=2.5,
                             cmap="plasma_r")

        # add a color legend
        cbar = fig1.colorbar(lines, ax=axs[1], label='Time (days)')
        cbar.ax.axhline(t0, c='limegreen', lw=3)

        # finish the phase portrait plot
        axs[1].set_xlabel("Person 1")
        axs[1].set_ylabel("Person 2")
        axs[1].set_title("Phase Portrait")
        axs[1].set_xlim([-10.5, 10.5])
        axs[1].set_ylim([-10.5, 10.5])
        plt.show()

    return sol.t, sol.y[0], sol.y[1]
```

Oscillating Personal Happiness

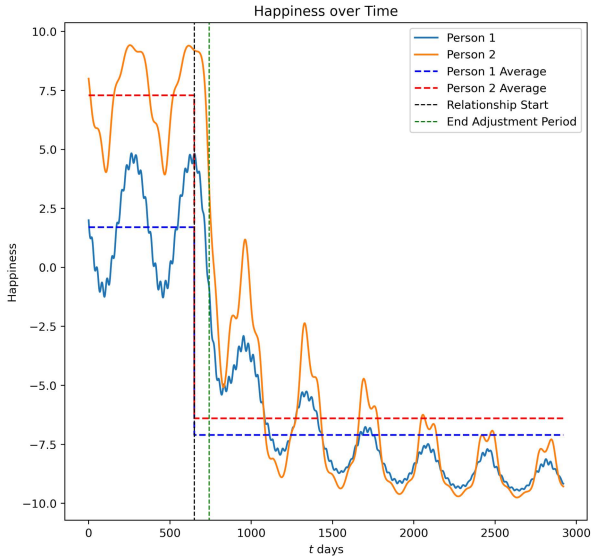
In [29]:
```
approx = base_model(x0=np.array([-0.2, 0.8]), gamma=np.array([0.01, 0.05]), amp=np.
                    d=np.array([2*np.pi / 30, 2*np.pi / 90]), week_amp=np.array([.0
                    t0=500, days_adjust=183, plot_avg_lines=True, plot_event_lines=
```

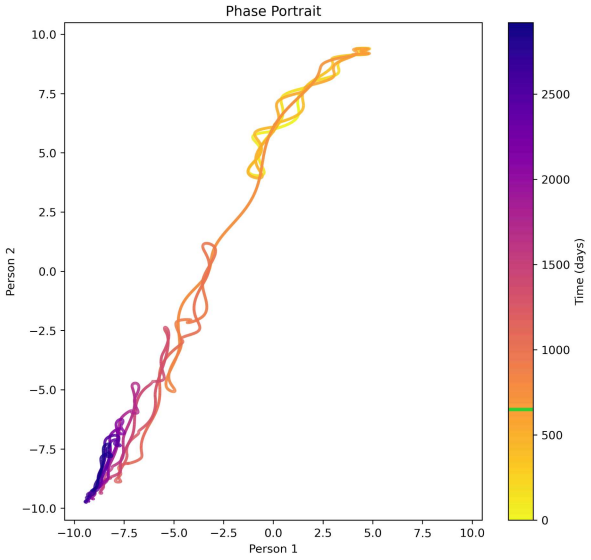


In [30]:
```
approx = base_model(x0=np.array([0.2, 0.8]), gamma=np.array([0.01, -0.05]), amp=np.
                    d=np.array([2*np.pi / 30, 2*np.pi / 90]), week_amp=np.array([.0
```



In [31]:
```
approx = base_model(x0=np.array([0.2, 0.8]), gamma=np.array([-0.01, 0.02]), amp=np.
                    d=np.array([2*np.pi / 30, 2*np.pi / 90]), week_amp=np.array([0.
```





The Time Variant Reactive functions

In [11]:
```
def base_model(x0=np.array([0.5, 0.5]), gamma=np.array([0.5, 0.5]), c=np.array([2*n
               amp=np.array([0, 0]), week_amp=np.array([0, 0]), t0=0, days_adjust=9
    '''Return the approximated solution to the two-person relationship model under
        1. The personal happiness of each individual is either constant or periodic
        2. The level of reactivity of each individual is measured by a constant.

        Parameters:
            x0 ((1,2) ndarray) : the initial conditions, i.e. starting happiness va
                - defaults to 0.5 for both people
            gamma ((1,2) array-like) : the reactivity coefficients for persons 1 an
                - defaults to 0.5 for both people
            c ((1, 2) array-like) : the period of individual happiness fluctuation
                - defaults to 2*np.pi / 365 (happiness fluctuates seasonally)
            amp ((1, 2) array-like) : dictates the amplitude of each person's indiv
                - defaults to (0, 0) (corresponding to constant invidiual happiness
            d ((1, 2) array-like) : the period of individual happiness fluctuation
                - defaults to 2*np.pi / 365 (happiness fluctuates seasonally)
            week_amp ((1, 2) array-like) : dictates the amplitude of each person's
                - defaults to (0, 0) (corresponding to constant invidiual happiness
            t0 (float) : the time at which persons 1 and 2 begin interacting
                - defaults to 0 (always in relationship)
            days_adjust (int) : the number of days for both people to adjust to rel
                - defaults to 90
            num_years (float) : the length of the time period to evaluate, in years
                - defaults to 1
            plot (bool) : whether to show solution plots in addition to returning t
                - defaults to True
            plot_avg_lines (bool) : whether to plot the average happiness lines on
                - defaults to True
            plot_event_lines (bool) : whether to plot the relationship event vertic
                - defaults to True

        Returns:
            (ndarray, ndarray, ndarray) : the approximated solution to the IVP
                - includes time steps and approximated values for person 1 and pers
```

```
    '''
    T = int(num_years * 365)
    t_span = (0, T)
    t_vals = np.linspace(0, T, 3*T + 1)

    # the indicator function determines whether the individuals are interacting
    ind = lambda t : int(t >= t0)
    tf = t0 + days_adjust

    # function that implements an adjustment period after the start of the relation
    def _adjust(t):
        if t < t0: return 0
        elif t > tf: return 1
        else: return 1 - (tf - t) / (tf - t0)

    # define the ODE
    def base_ode(t, x):
        # personal happiness components
        H1p_dot= -1 * (amp[0] * c[0] * np.cos(c[0] * t) + week_amp[0] * d[0] * np.c
        H2p_dot = -1 * (amp[1] * c[1] * np.cos(c[1] * t) + week_amp[1] * d[1] * np.

        # total happiness
        H1t_dot = H1p_dot + gamma[0](t) * (x[1] - x[0]) * ind(t) * _adjust(t)
        H2t_dot = H2p_dot + gamma[1](t) * (x[0] - x[1]) * ind(t) * _adjust(t)

        # return -- scale by 'carrying capacity' to keep solutions bounded
        return np.array([
            H1t_dot * (1 - x[0] / 1) * (1 + x[0] / 1),
            H2t_dot * (1 - x[1] / 1) * (1 + x[1] / 1)
        ])

    # approximate solution to IVP
    sol = solve_ivp(base_ode, t_span, x0, t_eval=t_vals)
    p1_vals = sol.y[0] * 10
    p2_vals = sol.y[1] * 10

    # plot solutions if desired
    if plot:
        # scale happiness levels to 0 - 10, like survey results
        fig1, axs = plt.subplots(nrows=1, ncols=2)
        axs[0] = plt.subplot(121)
        axs[0].plot(sol.t, p1_vals, label='Person 1')
        axs[0].plot(sol.t, p2_vals, label='Person 2')

        # plot the average lines and relationship event lines
        index = int(t0 * (3*T + 1) / T)
        if plot_avg_lines:
            axs[0].plot(sol.t, [np.mean(p1_vals[:index])] * index + [np.mean(p1_val
            axs[0].plot(sol.t, [np.mean(p2_vals[:index])] * index + [np.mean(p2_val
        if plot_event_lines:
            axs[0].axvline(t0, ymin=min(min(p1_vals), min(p2_vals)) - .5, ymax=max(
            axs[0].axvline(tf, ymin=min(min(p1_vals), min(p2_vals)) - .5, ymax=max(

        # finish the happiness plot
        axs[0].set_xlabel(r"$t$ days")
        axs[0].set_ylabel("Happiness")
```

```
        axs[0].legend()
        axs[0].set_title("Happiness over Time")

        # plot the phase portrait
        lines = colored_line(sol.y[0]*10, sol.y[1]*10, np.linspace(0, sol.t[-1], le

        # add a color legend
        cbar = fig1.colorbar(lines, ax=axs[1], label='Time (days)')
        cbar.ax.axhline(t0, c='limegreen', lw=3)

        # finish the phase portrait plot
        axs[1].set_xlabel("Person 1")
        axs[1].set_ylabel("Person 2")
        axs[1].set_title("Phase Portrait")
        axs[1].set_xlim([-10.5, 10.5])
        axs[1].set_ylim([-10.5, 10.5])
        plt.show()

    return sol.t, sol.y[0], sol.y[1]
```

In [ ]:
```
# person's reactivity increases to a certain capacity and stays
def pos_log(t):
    """Return the approximated solution to the two-person relationship model under
        person happiness is distributed with logistic function
        L is how much your reaction can varry
        s is how much of the reaction is negative
        k is steepness, rate of reaction"""
    L = 0.5
    k = 1 / 30
    s = 0.2
    g = 200
    return (L / (1 + np.exp(-k * (t - g))) - s)
```

In [ ]:
```
# person's reactivity decreases to a certain capacity and stays
def neg_log(t):
    """Return the approximated solution to the two-person relationship model under
        person happiness is distributed with negative logistic function
        L is how much your reaction can varry
        s is how much of the reaction is negative
        k is steepness, rate of reaction"""
    L = 0.5
    k = 1 / 30
    s = 0.2
    g = 200
    return (1 / 10) * (-L / (1 + np.exp(-k * (t - g))) - s)
```

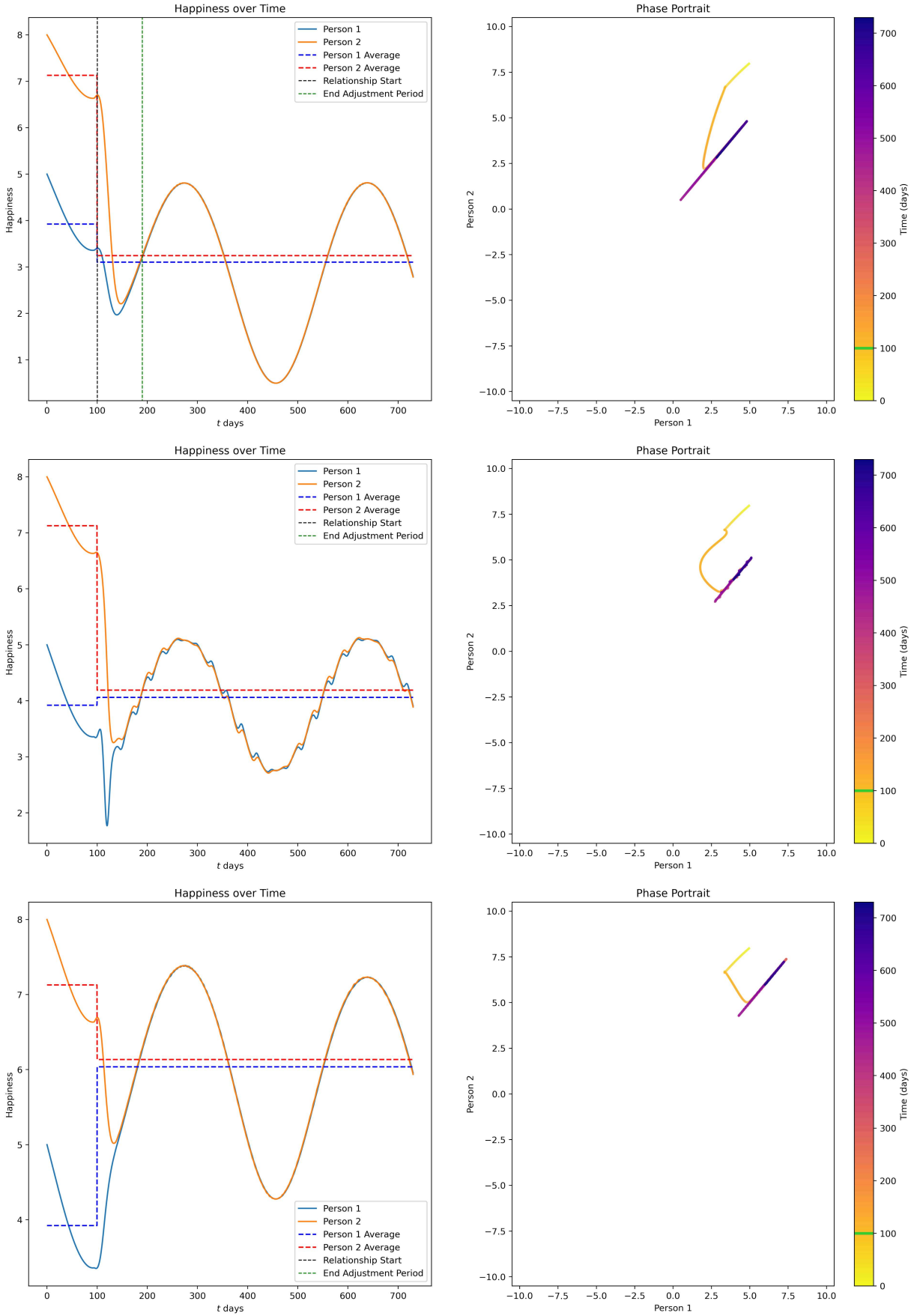
In [ ]:
```
# person's reactivity is constantly varying
def pos_sin(t):
    """Return the approximated solution to the two-person relationship model under
        person happiness is distributed with oscilation
        a is applitude
        b is period"""
    a = 0.5
    b = (2 * np.pi) / 30
    return (a * np.sin(b * t))
```

In [ ]:
```
# person's reactivity is constantly varying
def neg_sin(t):
    """Return the approximated solution to the two-person relationship model under
        person happiness is distributed with oscilation
        a is applitude
        b is period"""
    a = 0.5
    b = (2 * np.pi) / 30
    return (-a * np.sin(b * t))
```

In [ ]:
```
# person's reactivity increases until a certain level of relationship comfort and t
def pos_norm(t):
    """Return the approximated solution to the two-person relationship model under
        person happiness is normally distrubuted
        a is mean
        b is variance
        c is amplitude
        d is offset"""
    a = 300
    b = 50
    c = 0.5
    d = 0.2
    return 125 * c *(norm.pdf(t, a, b)) + d
```

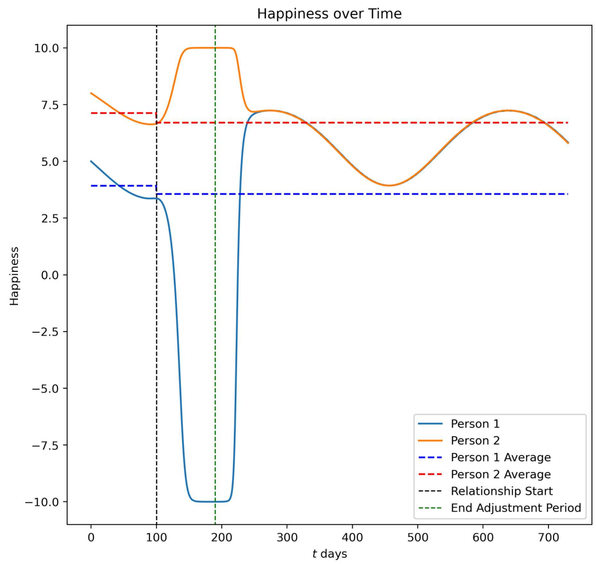
In [ ]:
```
# person's reactivity decreases until a certain level of relationship comfort and t
def neg_norm(t):
    """Return the approximated solution to the two-person relationship model under
        person happiness is negatively normally distrubuted
        a is mean
        b is variance
        c is amplitude
        d is offset"""
    a = 300
    b = 50
    c = 0.5
    d = 0.2
    return 125 * c * (-norm.pdf(t, a, b)) + d
```

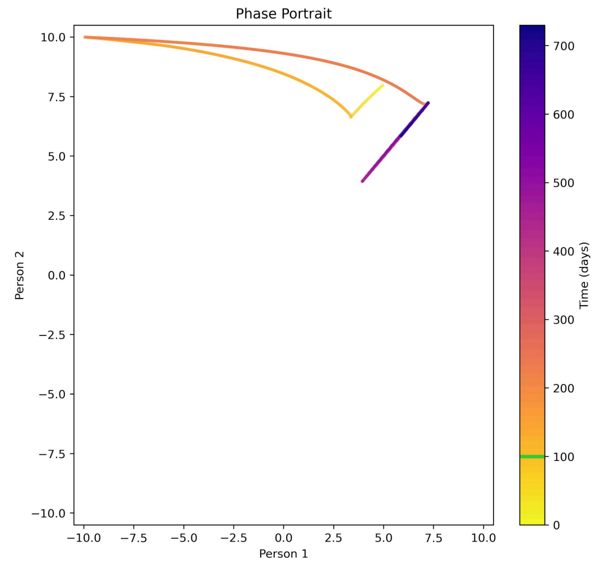
In [18]:
```
# plot all types of reactivity against a person with constant reactivity
approx_pos_log = base_model(x0=np.array([0.5,0.8]), gamma=(pos_log, lambda t: 0.5),
approx_sin = base_model(x0=np.array([0.5,0.8]), gamma=(pos_sin, lambda t: 0.5), amp
approx_pos_norm = base_model(x0=np.array([0.5,0.8]), gamma=(pos_norm, lambda t: 0.5
```

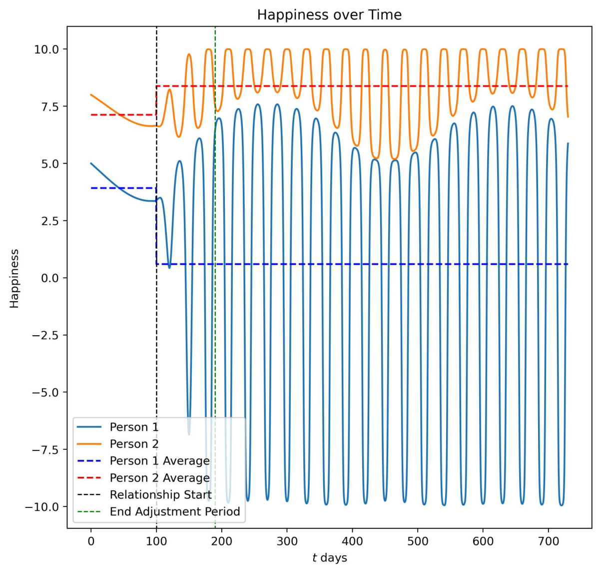


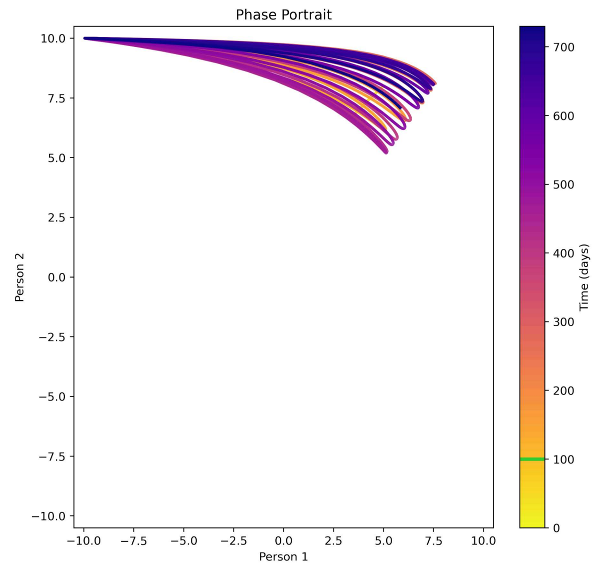
In [19]:
```
# plot 2 people with the same reactivity
approx_pos_log = base_model(x0=np.array([0.5,0.8]), gamma=(pos_log, pos_log), amp=n
```

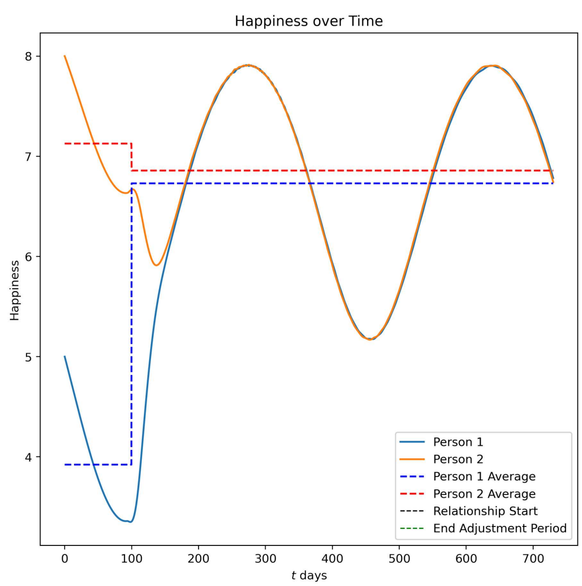
```
approx_sin = base_model(x0=np.array([0.5,0.8]), gamma=(pos_sin, pos_sin), amp=np.ar
approx_pos_norm = base_model(x0=np.array([0.5,0.8]), gamma=(pos_norm, pos_norm), am
```

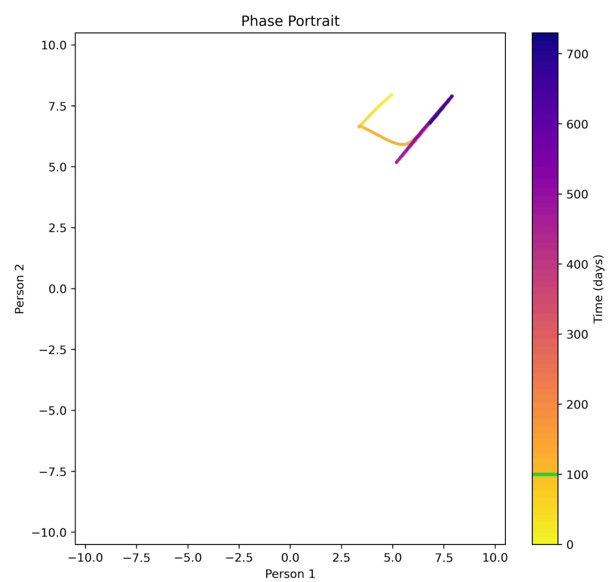




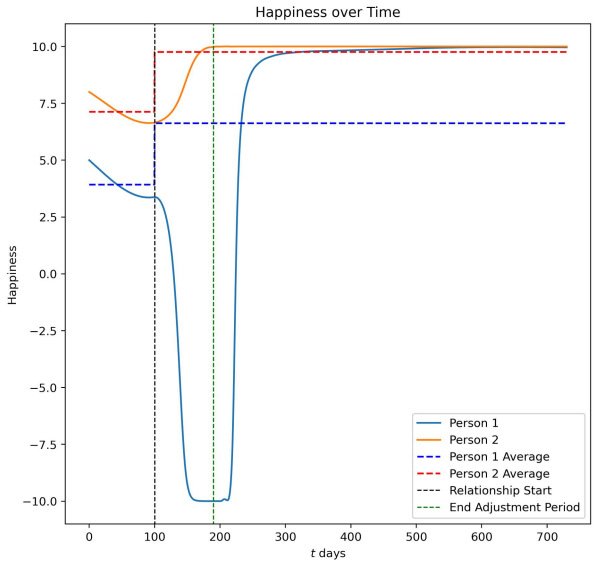


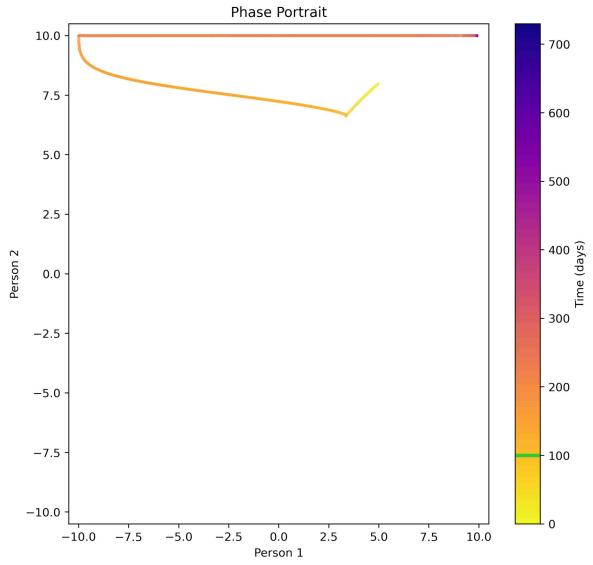


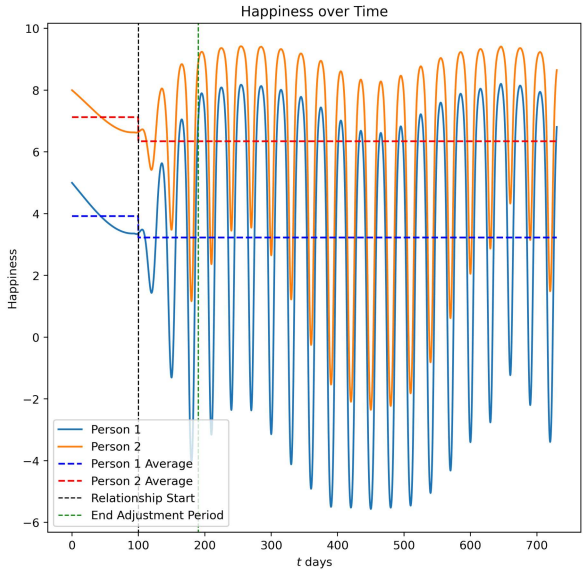


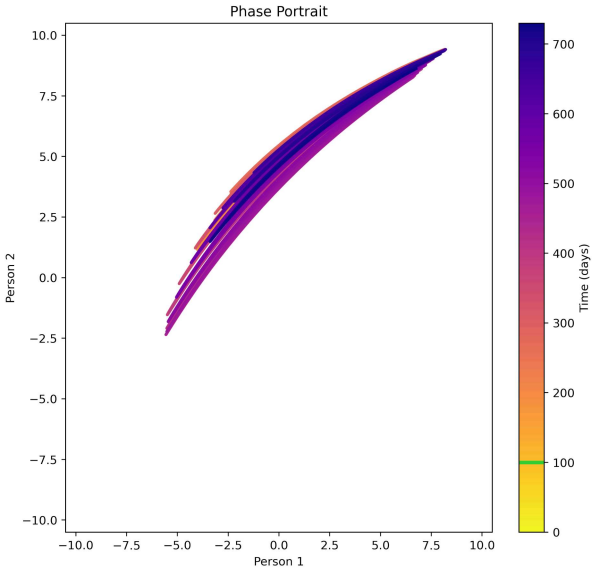


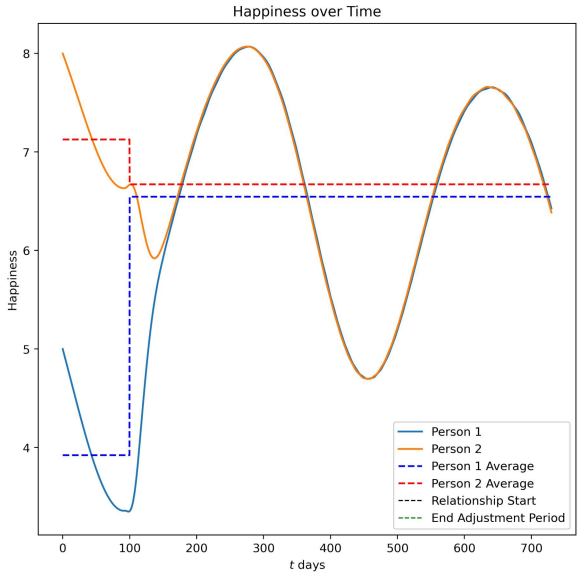
In [20]:
```
# plot people with exactly opposite reactivity
opposite_log = base_model(x0=np.array([0.5,0.8]), gamma=(pos_log, neg_log), amp=np.
opposite_sin = base_model(x0=np.array([0.5,0.8]), gamma=(pos_sin, neg_sin), amp=np.
opposite_norm = base_model(x0=np.array([0.5,0.8]), gamma=(pos_norm, neg_norm), amp=
```

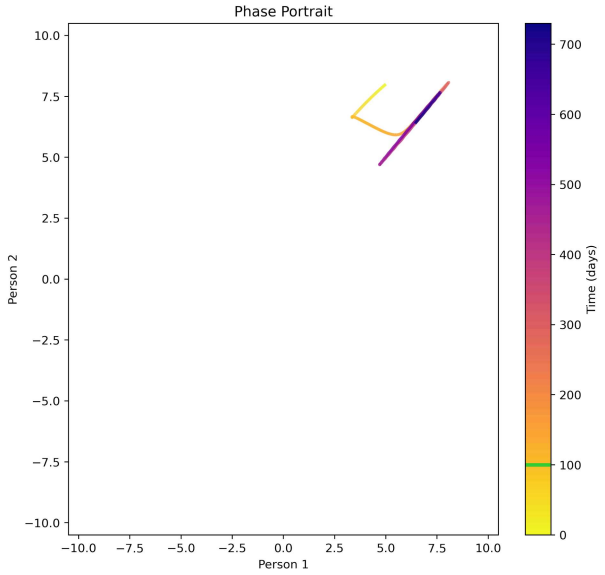






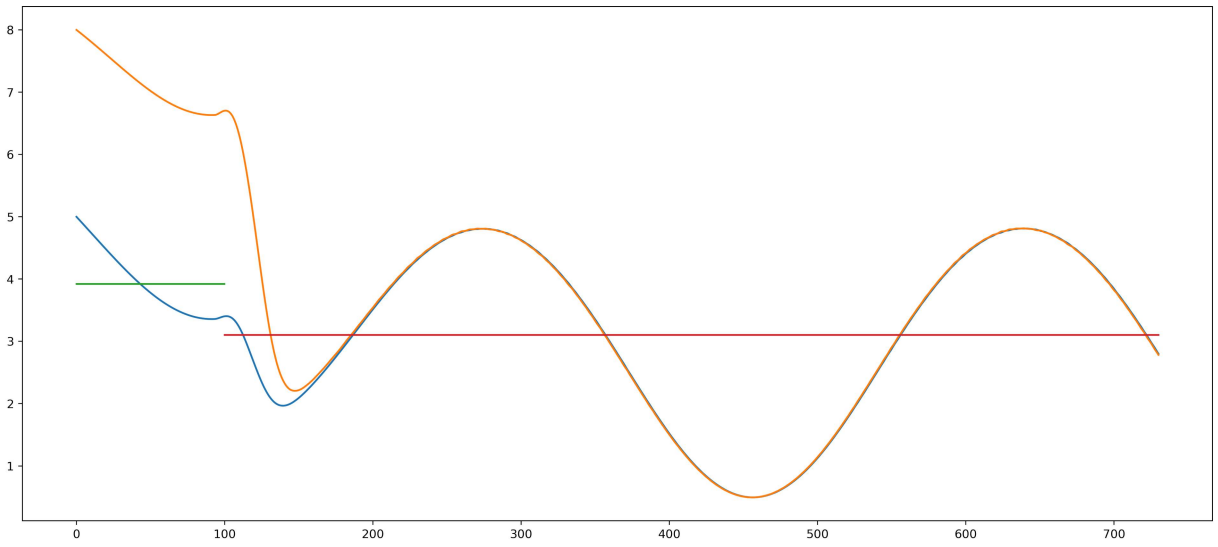






In [21]:
```
log1 = base_model(x0=np.array([0.5,0.8]), gamma=(pos_log, lambda t: 0.5), amp=np.ar
log2 = base_model(x0=np.array([0.5,0.8]), gamma=(pos_log, pos_log), amp=np.array([0
log3 = base_model(x0=np.array([0.5,0.8]), gamma=(pos_log, neg_log), amp=np.array([0
```

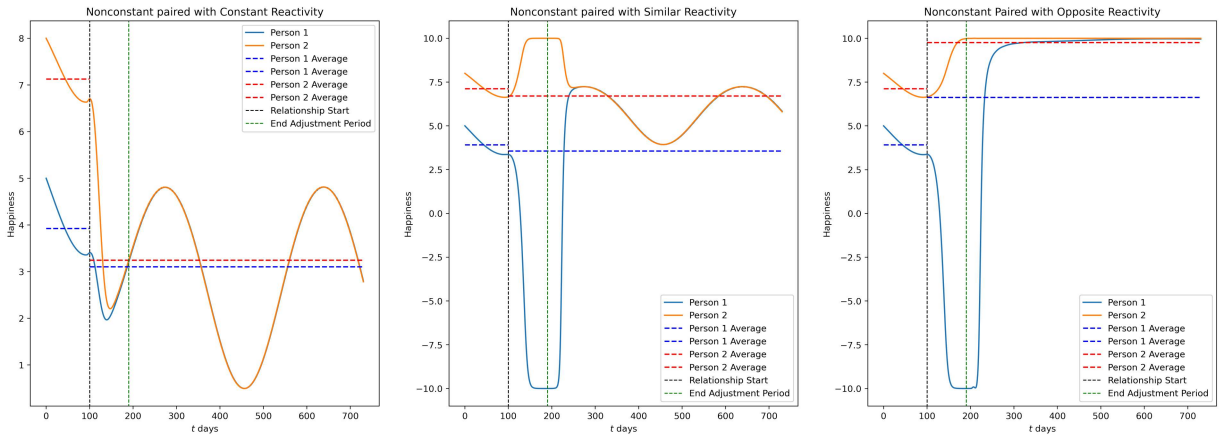
In [22]:
```
t, y1, y2 = log1
t0 = 100
T = t0 + 90
index = int(t0 * (3*T + 1) / T)

plt.plot(t, y1 * 10)
plt.plot(t, y2 * 10)
plt.plot(t[:index], [np.mean(y1[:index]) * 10] * index)
plt.plot(t[index:], [np.mean(y1[index:]) * 10] * (len(t) - index))
plt.show()
```



In [23]:
```
t0 = 100
T = t0 + 90
fig1, ax = plt.subplots(nrows=1, ncols=3)
fig1.set_figheight(8)
fig1.set_figwidth(24)
```

```
# first subplot
ax[0] = plt.subplot(131)
plt.plot(log1[0], log1[1] * 10, label="Person 1")
plt.plot(log1[0], log1[2] * 10, label="Person 2")


index = int(t0 * (3*T + 1) / T)

ax[0].plot(log1[0][:index], [np.mean(log1[1][:index] * 10)] * index, 'b--', label="
ax[0].plot(log1[0][index:], [np.mean(log1[1][index:] * 10)] * (len(log1[0]) - index
ax[0].plot(log1[0][:index], [np.mean(log1[2][:index] * 10)] * index, 'r--', label="
ax[0].plot(log1[0][index:], [np.mean(log1[2][index:] * 10)] * (len(log1[0]) - index

ax[0].axvline(t0, ymin=min(min(log1[1]), min(log1[2])) - .5, ymax=max(max(log1[1]),
ax[0].axvline(T, ymin=min(min(log1[1]), min(log1[2])) - .5, ymax=max(max(log1[1]),

ax[0].set_xlabel(r"$t$ days")
ax[0].set_ylabel("Happiness")
ax[0].legend()
ax[0].set_title("Nonconstant paired with Constant Reactivity")

# second subplot
ax[1] = plt.subplot(132)
plt.plot(log2[0], log2[1] * 10, label="Person 1")
plt.plot(log2[0], log2[2] * 10, label="Person 2")

ax[1].plot(log2[0][:index], [np.mean(log2[1][:index] * 10)] * index, 'b--', label="
ax[1].plot(log2[0][index:], [np.mean(log2[1][index:] * 10)] * (len(log2[0]) - index
ax[1].plot(log2[0][:index], [np.mean(log2[2][:index] * 10)] * index, 'r--', label="
ax[1].plot(log2[0][index:], [np.mean(log2[2][index:] * 10)] * (len(log2[0]) - index

ax[1].axvline(t0, ymin=min(min(log2[1]), min(log2[2])) - .5, ymax=max(max(log2[1]),
ax[1].axvline(T, ymin=min(min(log2[1]), min(log2[2])) - .5, ymax=max(max(log2[1]),

ax[1].set_xlabel(r"$t$ days")
ax[1].set_ylabel("Happiness")
ax[1].legend()
ax[1].set_title("Nonconstant paired with Similar Reactivity")

# third subplot
ax[2] = plt.subplot(133)
plt.plot(log3[0], log3[1] * 10, label="Person 1")
plt.plot(log3[0], log3[2] * 10, label="Person 2")

ax[2].plot(log3[0][:index], [np.mean(log3[1][:index] * 10)] * index, 'b--', label="
ax[2].plot(log3[0][index:], [np.mean(log3[1][index:] * 10)] * (len(log3[0]) - index
ax[2].plot(log3[0][:index], [np.mean(log3[2][:index] * 10)] * index, 'r--', label="
ax[2].plot(log3[0][index:], [np.mean(log3[2][index:] * 10)] * (len(log3[0]) - index

ax[2].axvline(t0, ymin=min(min(log3[1]), min(log3[2])) - .5, ymax=max(max(log3[1]),
ax[2].axvline(T, ymin=min(min(log3[1]), min(log3[2])) - .5, ymax=max(max(log3[1]),

ax[2].set_xlabel(r"$t$ days")
ax[2].set_ylabel("Happiness")
ax[2].legend()
```

```
ax[2].set_title("Nonconstant Paired with Opposite Reactivity")

plt.show()
```



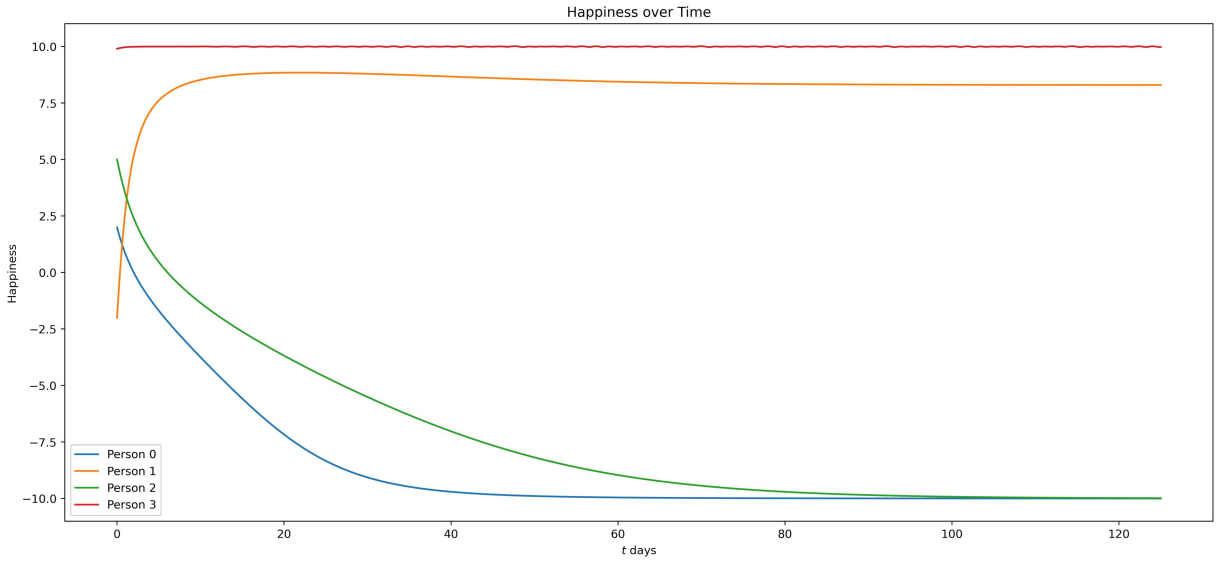
Group Model

In [ ]:

```
def generalized_model(x0=np.array([0.5, 0.5]),
                      α=np.array([0, 0]),
                      γ=np.array([[0.5, 0.5], [0.5, 0.5]]),
                      plot=True):
    '''Return the approximated solution to the two-person relationship model under
        1. The personal happiness of each individual is either constant or periodic
        2. The level of reactivity of each individual is measured by a constant.

        Parameters:
            x0 ((n,) ndarray) : the initial conditions, i.e. starting happiness val
                - defaults to 0.5 for both people
            γ ((n,n) array-like) : the reactivity coefficients for persons 1 and 2,
                - defaults to 0.5 for both people
            plot (bool) : whether to show solution plots in addition to returning t

        Returns:
            (ndarray, ndarray, ndarray) : the approximated solution to the IVP
                - includes time steps and approximated values for person 1 and pers
    '''
    n = x0.size
    c = 1 / 52          # assume happiness is measured on a week to week period. Ca
    tf = 125
    t_span = (0, tf)  # default time interval of 1000 days. Can be adjusted
    t_vals = np.linspace(0, tf, tf * 10)

    # define the ODE
    def base_ode(t, x):
        # total happiness
        X1, X2 = np.meshgrid(x, x)
        X = γ * (X1 - X2)

        # return -- scale by 'carrying capacity' to keep solutions bounded
        return (X.sum(axis=1) + α * np.sin(x * c)) * (1 - x**2)

    # approximate solution to IVP
```

```
    sol = solve_ivp(base_ode, t_span, x0, t_eval=t_vals)

    # plot solutions if desired
    if plot:
        # scale happiness levels to 0 - 10, like survey results
        for i in range(n):
            plt.plot(sol.t, sol.y[i]*10, label=f'Person {i}')

        plt.xlabel(r"$t$ days")
        plt.ylabel("Happiness")
        plt.legend()
        plt.title("Happiness over Time")
        plt.show()

    return sol.t, sol.y
```

In [ ]:
```
x0 = np.array([0.2, -0.2, 0.5, .99])
α=np.array([0.8, 0.4, 0, 0])
γ=np.array([
    [0, .1, -0.1, -0.1],
    [0, 0, .05, .5],
    [0, 0.2, 0, -0.2],
    [-0.5, -0.2, 0, 0]
])

t, y = generalized_model(x0, α, γ)

np.linalg.eig(γ)
```




```
EigResult(eigenvalues=array([-0.32759977+0.j        ,  0.10124225+0.33427192j,
        0.10124225-0.33427192j,  0.12511528+0.j        ]), eigenvectors=array([[-0.4
575957 +0.j        , -0.02579968+0.21019988j,
        -0.02579968-0.21019988j, -0.23758334+0.j        ],
       [ 0.59669178+0.j        , -0.70148313+0.j        ,
        -0.70148313-0.j        ,  0.55650775+0.j        ],
       [-0.56826502+0.j        ,  0.18812852+0.38044691j,
         0.18812852-0.38044691j,  0.79389507+0.j        ],
       [-0.33412568+0.j        , -0.16085231-0.50701691j,
        -0.16085231+0.50701691j,  0.05986574+0.j        ]]))
```