Systems Biology Group Assignment

Henrik Åhl, Iñigo Ayestaran, Ramya Rangan
hpa22@cam.ac.uk, ia327@cam.ac.uk, rr520@cam.ac.uk

May 19, 2017

Question I

A

For this system, we can compute the H matrix element-wise from the $H_{ij} = \left.\frac{\partial \ln\left(R_i^- / R_i^+\right)}{\partial \ln x_j}\right|_{x=\langle x\rangle}$ relationship. This gives us the following H matrix:

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

To then find the drift matrix, we have that $M_{ij} = \frac{1}{\tau_i} H_{ij}$, yielding

$$M = \begin{pmatrix} \frac{1}{\tau_1} & 0 \\ -\frac{1}{\tau_2} & \frac{1}{\tau_2} \end{pmatrix}.$$

For the diffusion matrix, we have that the off-diagonals are 0 since none of the reactions simultaneously alter the number of both x_1 and x_2 in the system. The D matrix is then given by

$$D = \begin{pmatrix} \frac{2}{\tau_1}\frac{\langle s_1\rangle}{\langle x_1\rangle} & 0 \\ 0 & \frac{2}{\tau_2}\frac{\langle s_2\rangle}{\langle x_2\rangle} \end{pmatrix}.$$

In this system, since both species are produced and degraded one unit at a time, the average step sizes are 1, and we get

$$D = \begin{pmatrix} \frac{2}{\tau_1}\frac{1}{\langle x_1\rangle} & 0 \\ 0 & \frac{2}{\tau_2}\frac{1}{\langle x_2\rangle} \end{pmatrix}.$$

We can make some simplifications to the above matrices by using Little's law and the flux balance, which both apply at stationarity. From Little's law, we have that $\tau_i = \frac{1}{\beta_i}$. We then have

$$M = \begin{pmatrix} \beta_1 & 0 \\ -\beta_2 & \beta_2 \end{pmatrix}.$$

From the flux balance $\langle R^+\rangle = \langle R^-\rangle$, we have:

$$\langle x_1\rangle = \frac{\lambda_1}{\beta_1}$$

$$\langle x_2\rangle = \frac{\lambda_2\langle x_1\rangle}{\beta_2} = \frac{\lambda_2\lambda_1}{\beta_2\beta_1}$$

This then gives us the following matrix for D:

$$D = \begin{pmatrix} \frac{2\beta_1^2}{\lambda_1} & 0 \\ 0 & \frac{2\beta_2^2\beta_1}{\lambda_2\lambda_1} \end{pmatrix}$$

B

We first compute the matrix $M\eta$ to be

$$\begin{pmatrix} \frac{1}{\tau_1} & 0 \\ -\frac{1}{\tau_2} & \frac{1}{\tau_2} \end{pmatrix} \begin{pmatrix} \eta_{11} & \eta_{12} \\ \eta_{12} & \eta_{22} \end{pmatrix} = \begin{pmatrix} \frac{\eta_{11}}{\tau_1} & \frac{\eta_{12}}{\tau_1} \\ -\frac{\eta_{11}}{\tau_2} + \frac{\eta_{12}}{\tau_2} & -\frac{\eta_{12}}{\tau_2} + \frac{\eta_{22}}{\tau_2} \end{pmatrix}.$$

We can then compute $M\eta + (M\eta)^T$:

$$\begin{pmatrix} \frac{\eta_{11}}{\tau_1} & \frac{\eta_{12}}{\tau_1} \\ -\frac{\eta_{11}}{\tau_2} + \frac{\eta_{12}}{\tau_2} & -\frac{\eta_{12}}{\tau_2} + \frac{\eta_{22}}{\tau_2} \end{pmatrix} + \begin{pmatrix} \frac{\eta_{11}}{\tau_1} & -\frac{\eta_{11}}{\tau_2} + \frac{\eta_{12}}{\tau_2} \\ \frac{\eta_{12}}{\tau_1} & -\frac{\eta_{12}}{\tau_2} + \frac{\eta_{22}}{\tau_2} \end{pmatrix} = \begin{pmatrix} 2\frac{\eta_{11}}{\tau_1} & -\frac{\eta_{11}}{\tau_2} + \frac{\eta_{12}}{\tau_2} + \frac{\eta_{12}}{\tau_1} \\ -\frac{\eta_{11}}{\tau_2} + \frac{\eta_{12}}{\tau_2} + \frac{\eta_{12}}{\tau_1} & -2\frac{\eta_{12}}{\tau_2} + 2\frac{\eta_{22}}{\tau_2} \end{pmatrix}$$

From here we need to solve for η such that $M\eta + (M\eta)^T = D$, or

$$\begin{pmatrix} 2\frac{\eta_{11}}{\tau_1} & -\frac{\eta_{11}}{\tau_2} + \frac{\eta_{12}}{\tau_2} + \frac{\eta_{12}}{\tau_1} \\ -\frac{\eta_{11}}{\tau_2} + \frac{\eta_{12}}{\tau_2} + \frac{\eta_{12}}{\tau_1} & -2\frac{\eta_{12}}{\tau_2} + 2\frac{\eta_{22}}{\tau_2} \end{pmatrix} = \begin{pmatrix} \frac{2}{\tau_1}\frac{1}{\langle x_1 \rangle} & 0 \\ 0 & \frac{2}{\tau_2}\frac{1}{\langle x_2 \rangle} \end{pmatrix}.$$

From the top left elements of this matrix, we have

$$\eta_{11} = \frac{1}{\langle x_1 \rangle}$$

From the off-diagonal elements of the matrix, we have

$$\begin{aligned} -\tau_1\eta_{11} + (\tau_1 + \tau_2)\eta_{12} &= 0 \\ -\frac{\tau_1}{\langle x_1 \rangle} + (\tau_1 + \tau_2)\eta_{12} &= 0 \\ \eta_{12} &= \frac{\tau_1}{(\tau_1 + \tau_2)}\frac{1}{\langle x_1 \rangle}. \end{aligned}$$

Lastly, from the bottom right elements we have

$$-\eta_{12} + \eta_{22} = \frac{1}{\langle x_2 \rangle}$$

$$\eta_{22} = \frac{1}{\langle x_2 \rangle} + \frac{\tau_1}{(\tau_1 + \tau_2)}\frac{1}{\langle x_1 \rangle}.$$

This formulation for the fluctuation balance arises from approximating R^- as a linear function. In this case, our answer is an exact statement, because our rates consist of linear functions.

C

From the above formula, to have small η_{22} we must have that $\langle x_2 \rangle$ is large, so the average number of x_2 is large. In addition, we must have that $\frac{\tau_1}{(\tau_1+\tau_2)\langle x_1 \rangle}$ is small. To have this, we must have a large $\langle x_1 \rangle$ compared to $\frac{\tau_1}{\tau_1+\tau_2}$. Either $\langle x_1 \rangle$ (the average number of x_1) can be very large, or $\frac{\tau_1}{\tau_1+\tau_2} = \frac{1}{1+\frac{\tau_2}{\tau_1}}$ can be very small, which occurs when $\tau_2 \gg \tau_1$ (when the average lifetime of x_2 is larger than the average lifetime of x_1).

D

We will try to express $\rho_{12} = \frac{\eta_{12}}{\sqrt{\eta_{22}\eta_{11}}}$ in terms of $\frac{CV_2}{CV_1} = \sqrt{\frac{\eta_{22}}{\eta_{11}}}$. Recall that

$$\eta_{12} = \eta_{22} - \frac{1}{\langle x_2 \rangle}.$$

Then ρ_{12} is

$$\rho_{12} = \frac{\eta_{12}}{\sqrt{\eta_{22}\eta_{11}}} = \frac{\eta_{22} - \frac{1}{\langle x_2 \rangle}}{\sqrt{\eta_{22}\eta_{11}}} = \sqrt{\frac{\eta_{22}}{\eta_{11}}}(1 - \frac{1}{\langle x_2 \rangle \eta_{22}}) = \frac{CV_2}{CV_1}(1 - \frac{1}{\langle x_2 \rangle \eta_{22}})$$

E

When $\frac{1}{\langle x_2 \rangle}$ is negligible compared to η_{22}, $\frac{1}{\langle x_2 \rangle \eta_{22}}$ goes to zero. Then $\rho_{12} \approx \frac{CV_2}{CV_1}$. Since the ratio of CVs $\frac{CV_1}{CV_2}$ is positive, we can plot the relationship of $\frac{CV_1}{CV_2}$ to ρ_{12} as follows:

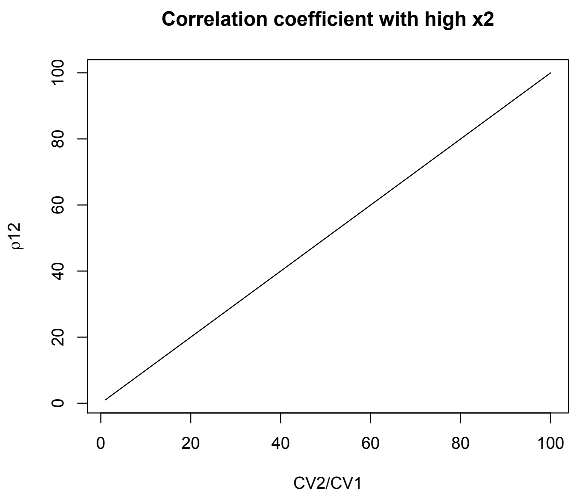

Figure 1: Theoretical relationship between ρ_{12} and $\frac{CV2}{CV1}$

F

In the high copy number limit, ρ_{12} can only take nonnegative values. This accessible range is reasonable for this system, as we would expect the correlation between x_1 and x_2 to be positive with x_1 increasing the production of x_2. With more mRNA molecules, more protein is produced.

Question II

A

In the Appendix, we include our code used to simulate the system from Question I with the Gillespie algorithm. For parameter values, we have $\langle x_2 \rangle = 1000$, $\tau_2 = 180 \times 60$, and τ_1 taking 100 values varying uniformly from 2×60 to 10×60. Our team's value for γ is $\gamma = round((29 + 17 + 30)/3) = 25$, yielding $\langle x_1 \rangle = 10 + 25 = 35$. Here, we round our value for γ to better resemble a biological system with discrete numbers of molecules. We can then compute β_i as $1/\tau_i$, and λ_i to ensure that the $\langle x_1 \rangle$ and $\langle x_2 \rangle$ values are as desired.

With these parameters, we ran the Gillespie algorithm by computing the probabilities of each reaction given the previous state of the system, choosing a reaction based on these probabilities, choosing a time delay based on the exponential distribution, and updating the time and state of the system as appropriate. We parallelized across τ_1 values in batches with a bash script.

B

To determine when our Gillespie simulation is finished, we can check that the flux balance and fluctuation balance equations hold. After 10^6 iterations, we begin calculating the relative errors in the flux balance and the relative deviations of η_{11}, η_{12}, and η_{22}. We run these calculations once every 10^5 iterations, computing statistics from timepoint 10^6 to the current time point. We chose to calculate statistics beginning at timepoint 10^6 to avoid biasing statistics with the initial fluctuations in the system, which were not at steady state. The simulation ended if all of these deviations are less than 10^{-3}, or if 20 million iterations had passed. In the appendix, we depict in Figure 7 the running averages of x_1 and x_2 through a simulation, and we show in Figure 8 the flux balance and fluctuation balance deviations through the simulation; these images indicate that the simulation has reached stationarity.

To calculate the flux and fluctuation balance deviations, we use a mean, variance, and covariance where the species' population at each time point is weighted by the time before the next reaction occurs. Say that the number of species of x_i at reaction j is x_{ij} and the time between reaction j and $j+1$ is t_j. For the weighted mean, we have

$$\langle x_i \rangle = \frac{\sum x_{ij} t_j}{\sum t_j}$$

For the weighted covariance, if $\langle x_i \rangle$ is the weighted mean, we have

$$Cov(x_1, x_2) = \frac{1}{\sum t_j} \sum (t_j (x_1 - \langle x_1 \rangle)(x_2 - \langle x_2 \rangle)).$$

The weighted variance of x_i is given by $Cov(x_i, x_i)$.

We use these formulations of the mean, covariance, and variance to check the flux and fluctuation balance. The relative error in fluxes is given by

$$\frac{\langle R^+ \rangle - \langle R^- \rangle}{\langle R^+ \rangle}.$$

If $\tilde{\eta}_{ij}$ are the expected normalized covariances as computed in Question I, the relative deviations of η_{ij} from their expected values is given by

$$\frac{\eta_{ij} - \tilde{\eta}_{ij}}{\tilde{\eta}_{ij}}$$

In Figures 2 and 3, we see the deviations for the flux balance and fluctuation balance at the end of the 100 simulations. We can see that values are centered at 0 taking quite low values, indicating that the simulations have reached stationarity. The relative errors in η_{12} and η_{22} are larger, perhaps due to the additional fluctuations in $\langle x_2 \rangle$.

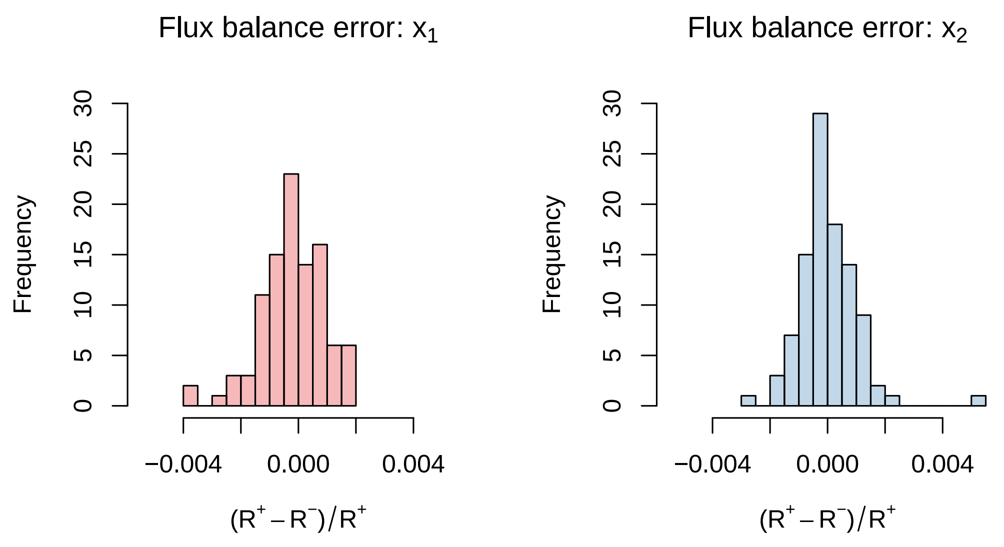

Figure 2: Flux balance

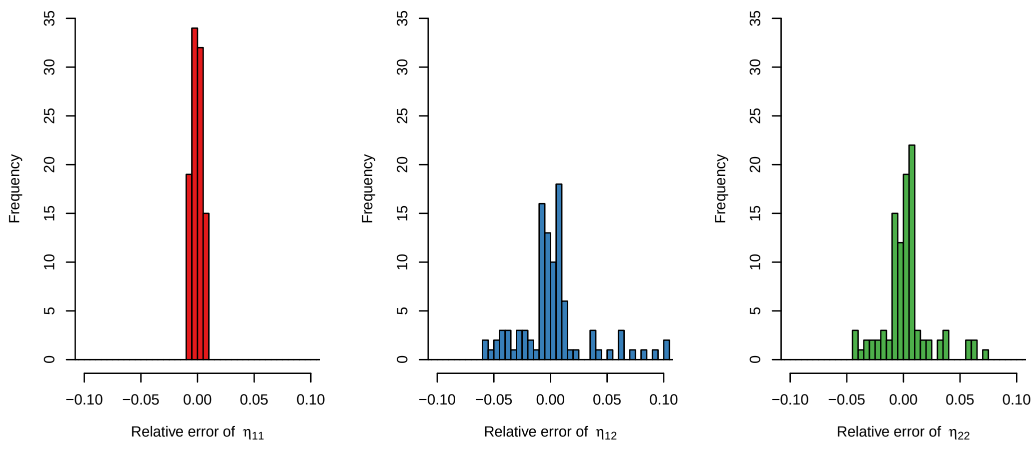

Figure 3: Fluctuation balance

C

In Figure 4, we depict the relationship between ρ_{12} and $CV2/CV1$ from theory and from simulations. The black line depicts the asymptotic relationship from Question I between ρ_{12} and $CV2/CV1$ when $\langle x_2 \rangle$ is large. On the left panel, η_{11}, η_{12}, and η_{22} are calculated from the theory in Question I with τ_1 varying from 2×60 to 10×60. In red, we superimpose the values $(\sqrt{\frac{\eta_{22}}{\eta_{11}}}, \frac{\eta_{12}}{\sqrt{\eta_{22}\eta_{11}}})$. In blue, we superimpose the values $(\sqrt{\frac{\eta_{22}}{\eta_{11}}}(1 - \frac{1}{\langle x_2 \rangle \eta_{22}}), \frac{\eta_{12}}{\sqrt{\eta_{22}\eta_{11}}})$. As we can see, the red values do not agree with the black line, indicating that fo $\langle x_2 \rangle = 1000$, $\frac{1}{\langle x_2 \rangle \eta_{22}}$ does not go to 0 and cannot be neglected. Only the blue values, which are an exact formula for the correlation coefficient, align with the black line.

We see this pattern from our simulations as well. In the right panel of Figure 4, we see the same points with η_{11}, η_{12}, and η_{22} computed from the simulation, yielding the same result: with $\langle x_2 \rangle =$

1000, the correlation coefficient ρ_{12} is not equal to $CV2/CV1$.

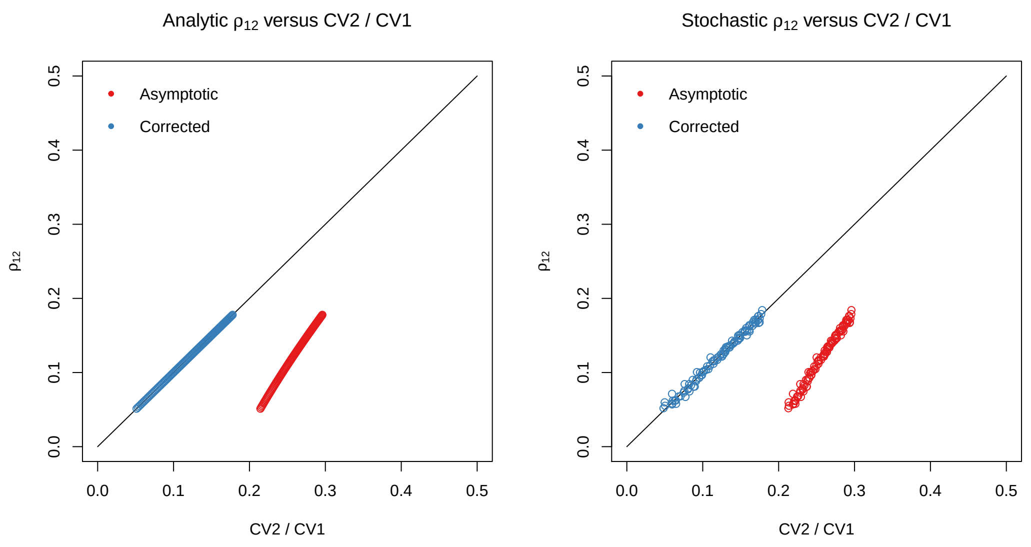

Figure 4: ρ_{12} and $\frac{CV2}{CV1}$: Predicted and simulated relationship

However, in Figure 5, we see that for larger values of $\langle x_2 \rangle$, the correlation coefficient begins to more closely approach the line $CV2/CV1$, with the red points approaching the blue points and black line. Here, we use the expected values of η_{11}, η_{12}, and η_{22} calculated in Question I. Indeed, with values of η_{22} close to 10^{-3}, $\langle x_2 \rangle$ must grow larger than 1000 before $\frac{1}{\langle x_2 \rangle \eta_{22}}$ goes to 0.

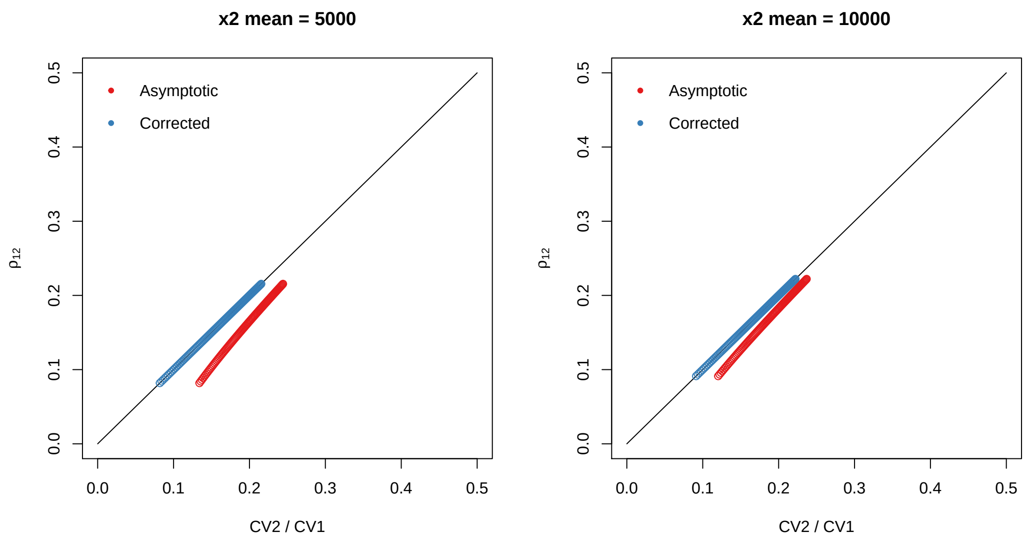

Figure 5: ρ_{12} and $\frac{CV2}{CV1}$: Approaching asymptotic relationship with higher $\langle x_2 \rangle$

D

In Figure 6 we can see the values for x_1 and x_2 over a short window of time in one of the simulated systems. At this time the system has already reached stationarity, so the average value (depicted by the dashed line) for either species is constant. We can see that the protein levels (x_2, in red) follow the fluctuations that happen in mRNA levels (x_1, in black).

There is a lag in how fast the protein levels adapt to the mRNA levels, given to differences in time constants. If we look at the definition of η_{12} in this system, the normalised covariance depends on $\frac{\tau_1}{\tau_1+\tau_2}$. In this system τ_1 is sensibly smaller than τ_2, meaning that mRNA molecules are more short-lived than proteins. Therefore, the covariance between the species is smaller as τ_1 is smaller.

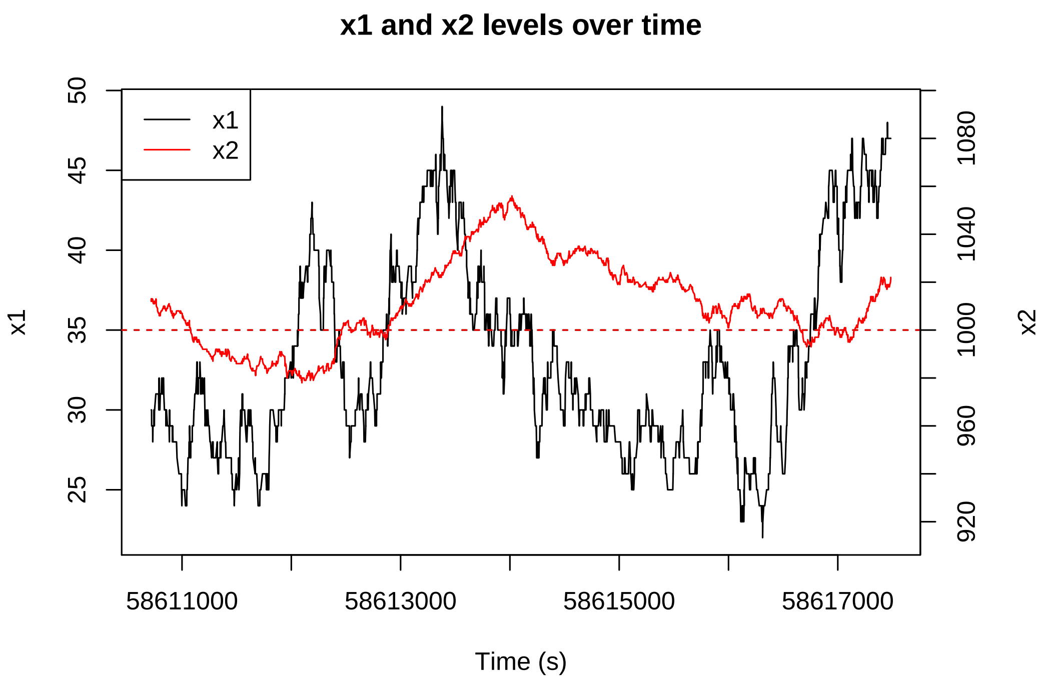

Figure 6: Fluctuations in x_1 and x_2 values over time. Dashed line indicates the average value for either species.

Appendix

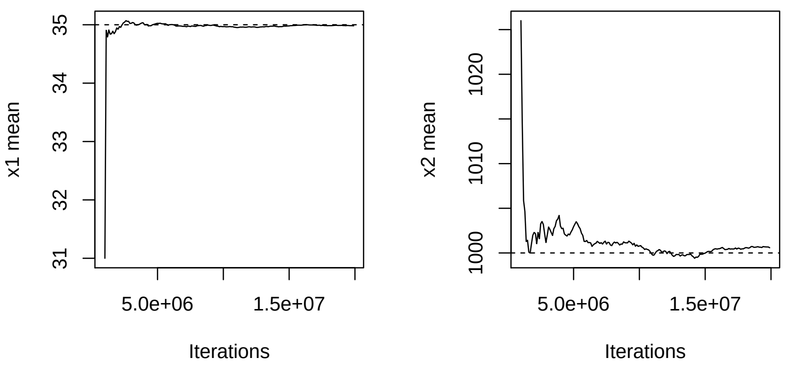

Figure 7: Running averages for both species over time. Dashed lines represent the long-term average values.

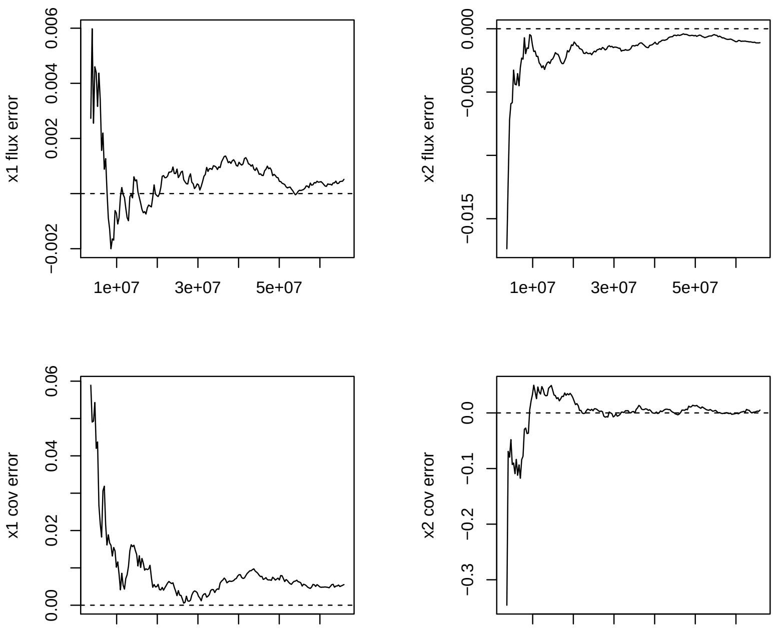

Figure 8: Relative errors for flux balances and fluctuation balances for either species. Dashed lines represent an error of 0.

Code

```sh
#!/bin/sh

## This script runs the simulations in R in parallel

run () {
  START=$1
  END=$2
  for ii in $(seq $START $END); do
    (
      RETURN_VAL=$(nice -n 5 Rscript --vanilla /local/data/public/hpa22/assignments/
          syba1/gillespie.R $ii)
    ) &
  done
}

run 1 20
wait
run 21 40
wait
run 41 60
wait
run 61 80
wait
run 81 100
```

gillespie.sh

```r
#!/usr/bin/env Rscript

# Import arguments from BASH script (choice of tau1)
args = commandArgs(trailingOnly=TRUE)

if (length(args)!=1) {
  stop("This script needs 1 argument: tau1 index.")
}

tau1.idx <- args[1]

# Simulation parameters
no.iterations = 20000000
no.reactions  = 4
no.species    = 2
tau1          = seq(2 * 60, 10 * 60, length.out = 100) # In seconds
epsilon       = 10e-3

# Iteration independent parameters
gamma    = round((29 + 17 + 30) / 3) # Birthdays
avg.x1   = 10 + gamma
avg.x2   = 1000
start.x1 = round(avg.x1 / 2)
start.x2 = round(avg.x2 / 2)
start.t  = 0
tau2     = 180 * 60
beta2    = 1 / tau2
lambda2  = (avg.x2 * beta2) / avg.x1

# Function to calculate a weighted (co)variance
weighted.cov = function(var1, var2 = var1, weights, w.mean1, w.mean2 = w.mean1){
  1/sum(weights) * sum(weights * (var1 - w.mean1) * (var2 - w.mean2))
}

# Main function
gillespie = function(tau1, no.iterations = no.iterations){
  # System parameters
  beta1 = 1 / tau1
  lambda1 = avg.x1 * beta1

  # Preallocation and definition of output matrix
```

```r
probabilities = rep(NA, no.reactions)
out.values = matrix(ncol = 1 + no.species + 2 + 3 + 3, nrow = no.iterations)
out.values[1, ] = c(start.t, start.x1, start.x2, rep(NA, 8))
ii = 2
while(ii < no.iterations + 1) {
  # Define reaction probabilities
  reactions = c(lambda1,
                beta1   * out.values[ii - 1, 2],
                lambda2 * out.values[ii - 1, 2],
                beta2   * out.values[ii - 1, 3])
  sum.reactions = sum(reactions)
  reactions = reactions / sum.reactions

  # Increment / decrement reacting species, choosing one of the 4 possible reactions
  reaction.sign = c(1, -1, 1, -1)
  reaction.index = sample(1:no.reactions, 1, prob=reactions)
  reaction.species = 1:no.species == ceiling(reaction.index / 2)

  # Update the chosen species
  out.values[ii, 2:3] = out.values[ii-1, 2:3] + reaction.sign[reaction.index] *
      reaction.species # indexing is slow

  # Update reaction time
  out.values[ii, 1] = out.values[ii - 1, 1] + rexp(1, sum.reactions)

  # Compute running average of each species, every 100000 iterations
  step.size = 100000
  start.pt = 1000000
  if (ii > start.pt & (ii - 1) %% step.size == 0) {
    x1s = out.values[start.pt:(ii-1), 2]
    x2s = out.values[start.pt:(ii-1), 3]
    wts = diff(out.values[start.pt:ii,1])
    # Weighted mean depending on different reaction times
    x1.mean = weighted.mean(x1s, wts)
    x2.mean = weighted.mean(x2s, wts)

    # Flux balance
    birth.x1 = lambda1
    death.x1 = beta1   * x1.mean
    birth.x2 = lambda2 * x1.mean
    death.x2 = beta2   * x2.mean

    # Return the relative errors in balance
    out.values[ii, 4] = (birth.x1 - death.x1) / birth.x1
    out.values[ii, 5] = (birth.x2 - death.x2) / birth.x2

    # Fluctuation balance (from simulation and expected)
    n.11 = weighted.cov(var1 = x1s, weights = wts,
                        w.mean1 = x1.mean) / (x1.mean ** 2)
    n.11.exp = 1 / x1.mean
    n.12 = weighted.cov(var1 = x1s, var2 = x2s, weights = wts,
                        w.mean1 = x1.mean,
                        w.mean2 = x2.mean) / (x1.mean * x2.mean)
    n.12.exp = tau1 / ((tau1 + tau2) * x1.mean)
    n.22 = weighted.cov(var1 = x2s, weights = wts,
                        w.mean1 = x2.mean) / (x2.mean ** 2)
    n.22.exp = 1 / x2.mean + n.12.exp

    # Return the relative errors in balance
    out.values[ii, 6] = (n.11 - n.11.exp) / n.11.exp
    out.values[ii, 7] = (n.12 - n.12.exp) / n.12.exp
    out.values[ii, 8] = (n.22 - n.22.exp) / n.22.exp

    # Return normalised covariances
    out.values[ii, 9:11] = c(n.11, n.12, n.22)

    # Check if the errors are smaller than a threshold, and stop the simulation if
        true
    if(all(abs(out.values[ii, 4:8]) < epsilon)) {
      return(out.values)
      }
    }
    ii = ii + 1
  }
  return(out.values)
}

# Function to plot the number of species over time
plot.num.species = function(a) {
  par(mfrow = c(1,2))
  plot(a[,1], a[,2], type='l', xlab = 'Time', ylab = 'x1')
  plot(a[,1], a[,3], type='l', xlab = 'Time', ylab = 'x2')
}

# Function to plot the relative errors in flux and covariances
plot.relative.err = function(a) {
  error.rows = !is.na(a[,4]) & is.finite(a[,6])
  par(mfrow = c(2,2), mai = c(.5,.7,.5,.5))
  plot(a[error.rows,1], a[error.rows,4], type='l', xlab='Time',
       ylab='x1 flux error', ylim = c(min(0, min(a[error.rows,4])),
                                     max(0,max(a[error.rows,4]))))
  abline(h = 0, lty = 2)
  plot(a[error.rows,1], a[error.rows,5], type='l', xlab='Time',
       ylab='x2 flux error', ylim = c(min(0, min(a[error.rows,5])),
                                     max(0,max(a[error.rows,5]))))
  abline(h = 0, lty = 2)
  plot(a[error.rows,1], a[error.rows,6], type='l', xlab='Time',
       ylab='x1 cov error', ylim =  c(min(0, min(a[error.rows,6])),
                                     max(0,max(a[error.rows,6]))))
  abline(h = 0, lty = 2)
  plot(a[error.rows,1], a[error.rows,8], type='l', xlab='Time',
       ylab='x2 cov error', ylim =  c(min(0, min(a[error.rows,8])),
                                     max(0,max(a[error.rows,8]))))
  abline(h = 0, lty = 2)
}

# Function to plot running mean values
plot.running.avg = function(a) {
  start.means = 1000000
  plot.vals = seq(start.means, no.iterations -1, by = 100000)
  x1.vals = sapply(plot.vals, function(x) weighted.mean(a[start.means:x,2],
                                                         diff(a[start.means:(x+1),1])))
  x2.vals = sapply(plot.vals, function(x) weighted.mean(a[start.means:x,3],
                                                         diff(a[start.means:(x+1),1])))
  par(mfrow=c(1,2))
  plot(plot.vals, x1.vals , type='l', xlab="Iterations",
       ylab="x1 mean")
  abline(h = avg.x1, lty = 2)

  plot(plot.vals, x2.vals,  type='l', xlab="Iterations",
       ylab="x2 mean")
  abline(h = avg.x2, lty = 2)
}

# Run the simulation
a <- gillespie(tau1[as.numeric(tau1.idx)], no.iterations=no.iterations)

# Store the resulting matrix in an .RData object
save(a, file = paste0(tau1.idx, "_result.RData"))
## All RData files save a matrix named "a", so loading any file
# will import an "a" matrix (watch out for overwriting!)
```

gillespie.R

```r
## This scripts import the results of the parallel simulations and
## extracts the data corresponding to the last iteration with data
## about flux and covariance error (when the system is stationary)
## Then we analyse and plot the results

setwd("/local/data/public/hpa22/assignments/syba1")
library(RColorBrewer)
```

```r
palette(brewer.pal(n = 8, name = "Set1"))

## Get data
no.files = 100
files = list.files()
files = files[grepl("result.RData", files)]
data = matrix(NA, nrow=no.files, ncol=11)
for(ii in 1:no.files){
  load(files[ii]) # loads matrix "a"
  error.rows = which(!is.na(a[,4]))
  data[ii, ] = a[error.rows[length(error.rows)], ] # store last row
}
last.rows = data # just rename

######## PART II.B Histograms

data = data.frame(last.rows)
colnames(data) = c("t", "x1", "x2", "flux.x1", "flux.x2",
                   "rel.n11", "rel.n12", "rel.n22",
                   "n11", "n12", "n22") # flux is in units of birth.x
attach(data)

# Histogram of relative errors in flux balance
par(mfrow = c(1, 2))
no.breaks = 20
shading = 1
hist(flux.x1, breaks=no.breaks, ylim=c(0,30), xlim = c(-0.0055, 0.0055),
     main = expression(paste("Flux balance error: ", x[1])),
     col = alpha(1, shading),
     xlab =  bquote('(R'^'+'-'R'^'-'*')'/ 'R'^'+'))

hist(flux.x2, breaks=no.breaks, ylim=c(0,30), xlim = c(-0.0055, 0.0055),
     main = expression(paste("Flux balance error: ", x[2])),
     col = alpha(2, shading),
     xlab =  bquote('(R'^'+'-'R'^'-'*')'/ 'R'^'+'))

# Histogram of relative errors in normalised covariances
par(mfrow=c(1, 3))
no.breaks = 20
hist(
  rel.n11,
  breaks = seq(-0.13, 0.13, by = 0.005),
  ylim = c(0, 35),
  xlim = c(-0.1, 0.1),
  main = "",
  col = alpha(1, shading),
  xlab =
    bquote("Relative error of " ~ eta[11])
)
hist(
  rel.n12,
  breaks = seq(-0.13, 0.13, by = 0.005),
  ylim = c(0, 35),
  xlim = c(-0.1, 0.1),
  main = "",
  col = alpha(2, shading),
  xlab = bquote("Relative error of " ~ eta[12])
)
hist(
  rel.n22,
  breaks = seq(-.13, .13, by = 0.005),
  ylim = c(0, 35),
  xlim = c(-0.1, 0.1),
  main = "",
  col = alpha(3, shading),
  xlab =
    bquote("Relative error of " ~ eta[22])
)


```

```
####### PART II.C Correlation coefficient (rho) vs CV2/CV1

# Function to plot the mathematically expected results (depending on x2 mean)
plot.analytic = function(x2.mean) {
  x1.mean = 35
  tau1 = seq(2*60, 10*60, length.out = 100)
  n11s = 1 / x1.mean
  n12s = tau1 / ((tau1 + 180 * 60) * x1.mean)
  n22s = 1 / x2.mean + n12s
  main.str = paste("x2 mean =", x2.mean)
  if (x2.mean == 1000) {
    main.str = expression(paste("Analytic ",
                                rho[12],
                                " versus CV2 / CV1"))
  }
  plot(sqrt(n22s/n11s), n12s/sqrt(n11s * n22s),
       xlim=c(0,0.5), ylim=c(0,0.5), col=1, xlab="CV2 / CV1",
       ylab=expression(rho[12]),
       main=main.str)
  lines(seq(0,0.5, 0.001), seq(0,0.5, 0.001))
  points(sqrt(n22s/n11s) * (1 - 1/(x2.mean * n22s)),
         n12s/sqrt(n11s * n22s), col=2)
  legend("topleft",
         legend=c("Asymptotic", "Corrected"),
         col=c(1, 2), pch=20, bty='n')
}

# Plot expected results with high x2 mean values
par(mfrow=c(1,2))
plot.analytic(5000)
plot.analytic(10000)

# Plot expected result with the parameter  used for simulations
par(mfrow=c(1,2))
plot.analytic(1000)

# Plot the result obtained from the simulation
n11s = last.rows[,9]
n12s = last.rows[,10]
n22s = last.rows[,11]
plot(sqrt(n22s/n11s), n12s/sqrt(n11s * n22s),
     xlim=c(0,0.5), ylim=c(0,0.5), col=1, xlab="CV2 / CV1",
     ylab=expression(rho[12]),
     main=expression(paste("Stochastic ",
                           rho[12], " versus CV2 / CV1")))
lines(seq(0,0.5, 0.001), seq(0,0.5, 0.001))
points(sqrt(n22s/n11s) * (1 - 1/(1000 * n22s)),
       n12s/sqrt(n11s * n22s), col=2)
legend("topleft",
       legend=c("Asymptotic", "Corrected"),
       col=c(1, 2), pch=20, bty='n')

########## PART II.D Visualization of both species over time

# Choose a interval from one of the simulations ("a" matrix)
load('100_result.RData')
i <- 17677000
interval <- i:(i+2000)

# We need to rescale the values of x2 so that the mean values of
# both variables are at the same height.

# Function to rescale x2
convert.y <- function(y1a, y1b, y2a, y2b, y, to=1) {
  slope <- (y1b - y1a) / (y2b - y2a)
  intercept <- y1a - slope * y2a
  if(to==1) intercept + slope * y else (y - intercept) / slope
}

# Rescale all values and mean value
```

```r
scaled.x2 <- convert.y(20, 50, 900, 1100, a[interval,3])
scaled.mean.x2 <- convert.y(20, 50, 900, 1100, 1000)

# Plot the values
par(mfrow = c(1,1), mar = c(5, 4, 4, 4) + 0.3)
plot(a[interval,1], a[interval,2], type='l', xlab = 'Time (s)', ylab = 'x1',
     main = 'x1 and x2 levels over time')
abline(h = avg.x1, lty = 2)
lines(a[interval,1], scaled.x2, col = 'red',
      bty = "n", xlab = "", ylab = "")
abline(h = scaled.mean.x2, lty = 2, col = 'red')

# Add x2 axis and legend
tick.labels <- seq(900, 1100, by = 20)
tick.locations <-  convert.y(20, 50, 900, 1100, tick.labels)
axis(4, at=tick.locations, labels=tick.labels)
mtext("x2", side=4, line=3)

legend('topleft', col = c('black','red'), lty = 1,
       legend = c('x1', 'x2'))
```

corr_cv.R