Fisheries management in random environments: comparison of harvesting policies for the logistic model

Supplementary material

Nuno M. Brites[1] (corresponding author brites@uevora.pt)

Carlos A. Braumann[1,2](braumann@uevora.pt)

July 17, 2017

[1]Centro de Investigação em Matemática e Aplicações, Instituto de Investigação e Formação Avançada, Universidade de Évora, Évora, 7000-671, Portugal

[2]Departamento de Matemática, Escola de Ciências e Tecnologia, Universidade de Évora, Évora, 7000-671, Portugal

1 Supplementary figures showing the time evolution of population size, effort and profit per unit time for scenarios S_1 to S_{14}

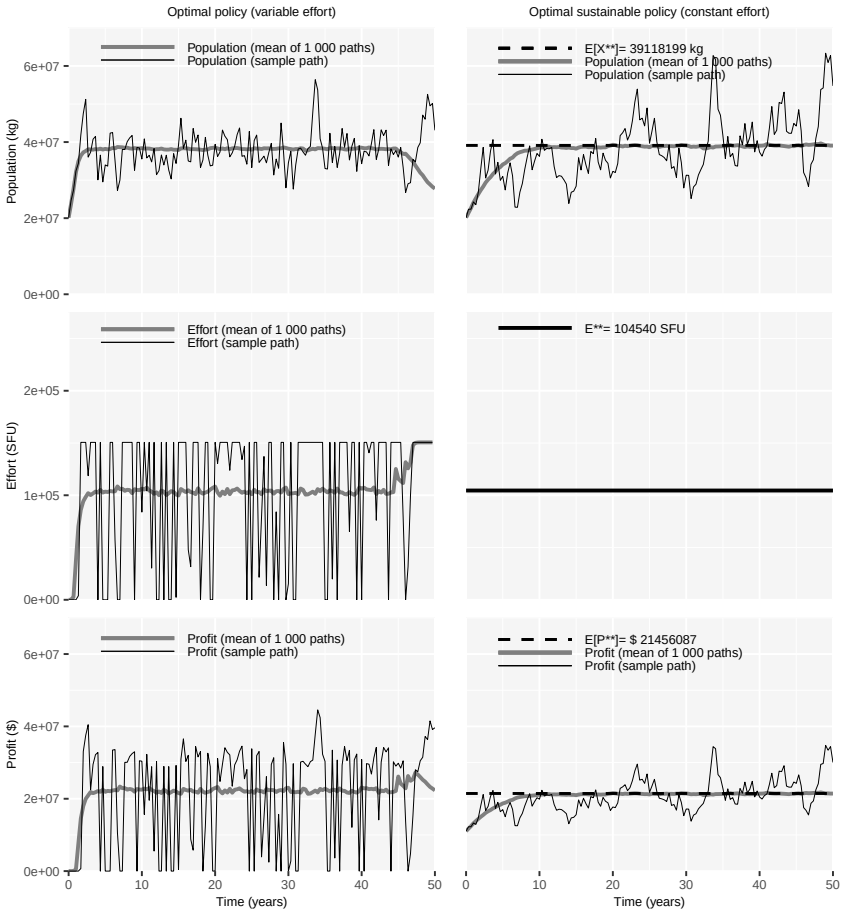

Figure 1: Scenario S_1: mean and randomly chosen sample path for the population, the effort and the profit per unit time. The optimal variable effort policy is on the left side and the optimal constant effort sustainable policy is on the right side.

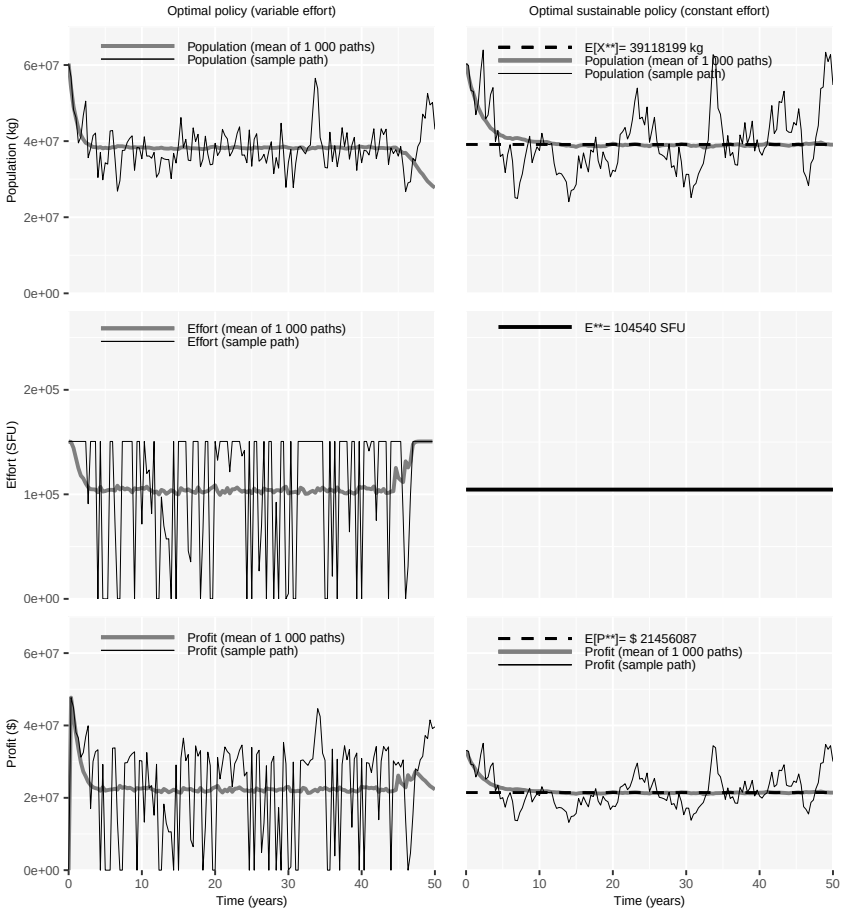

Figure 2: Scenario S_2: mean and randomly chosen sample path for the population, the effort and the profit per unit time. The optimal variable effort policy is on the left side and the optimal constant effort sustainable policy is on the right side.

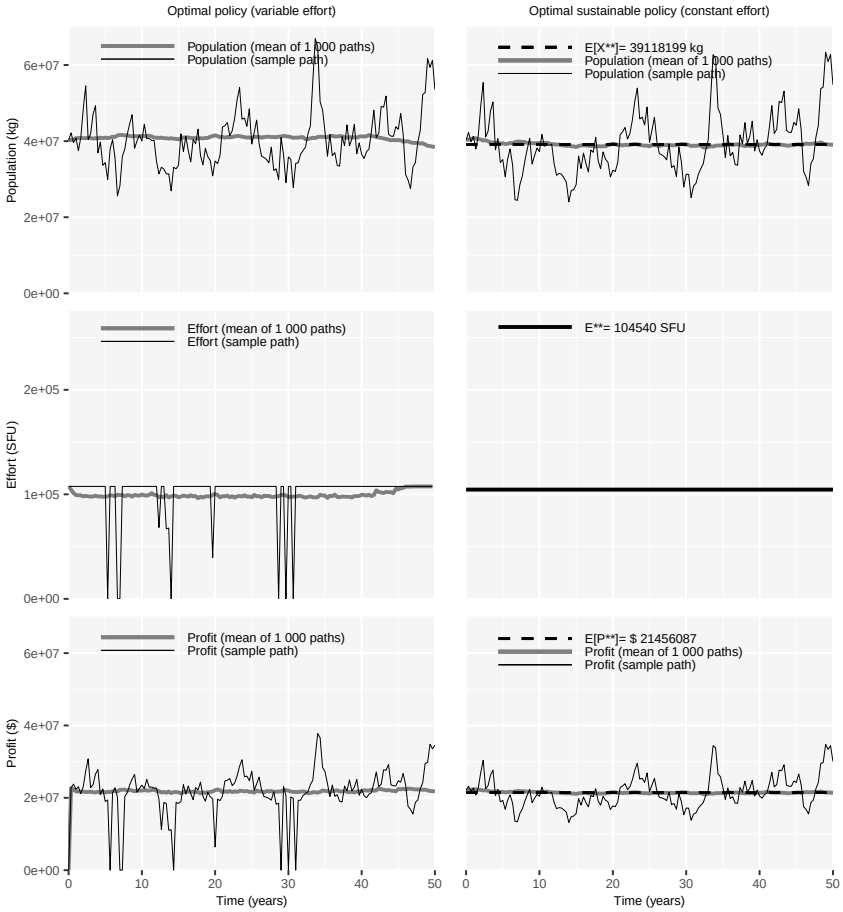

Figure 3: Scenario S_3: mean and randomly chosen sample path for the population, the effort and the profit per unit time. The optimal variable effort policy is on the left side and the optimal constant effort sustainable policy is on the right side.

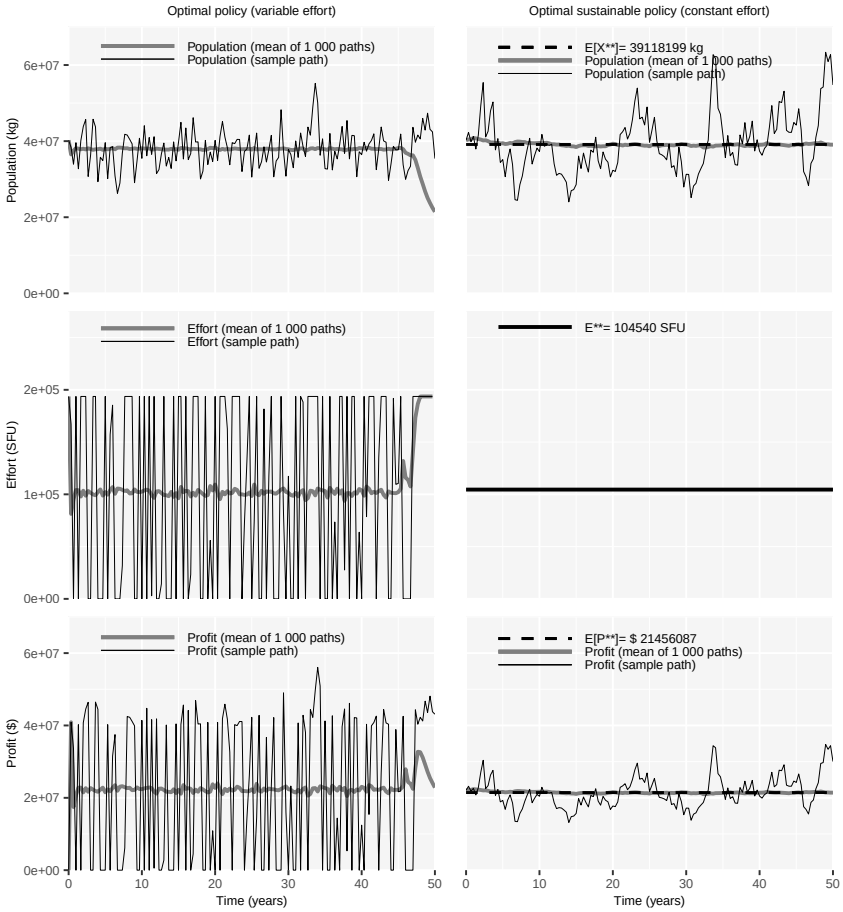

Figure 4: Scenario S_4: mean and randomly chosen sample path for the population, the effort and the profit per unit time. The optimal variable effort policy is on the left side and the optimal constant effort sustainable policy is on the right side.

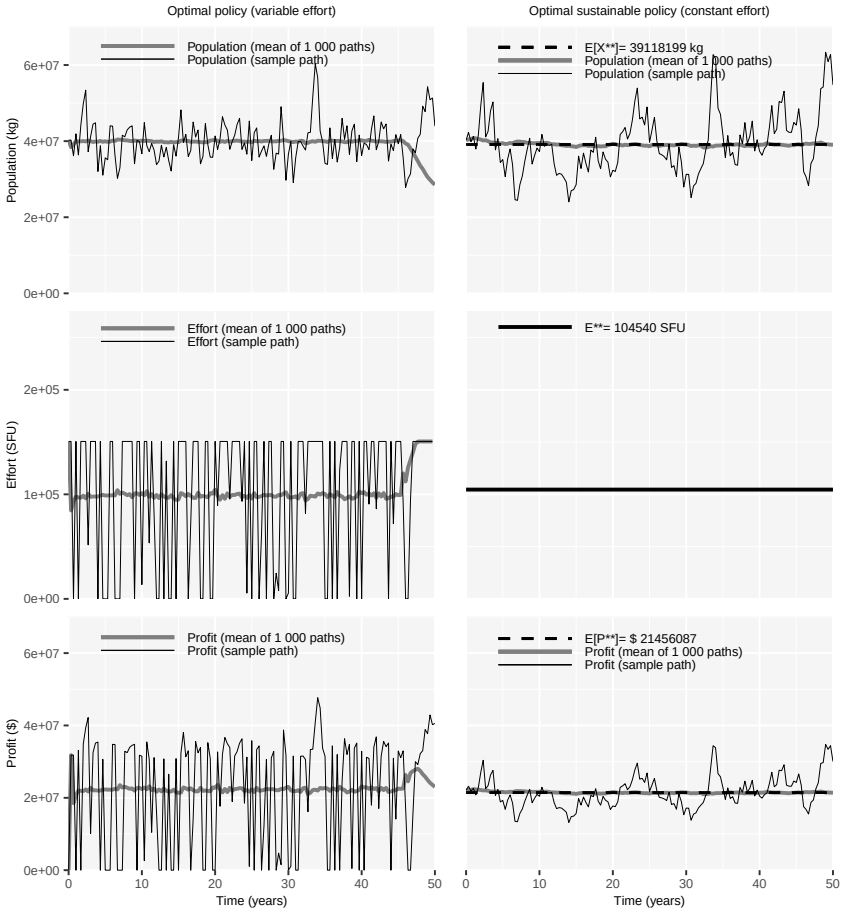

Figure 5: Scenario S_5: mean and randomly chosen sample path for the population, the effort and the profit per unit time. The optimal variable effort policy is on the left side and the optimal constant effort sustainable policy is on the right side.

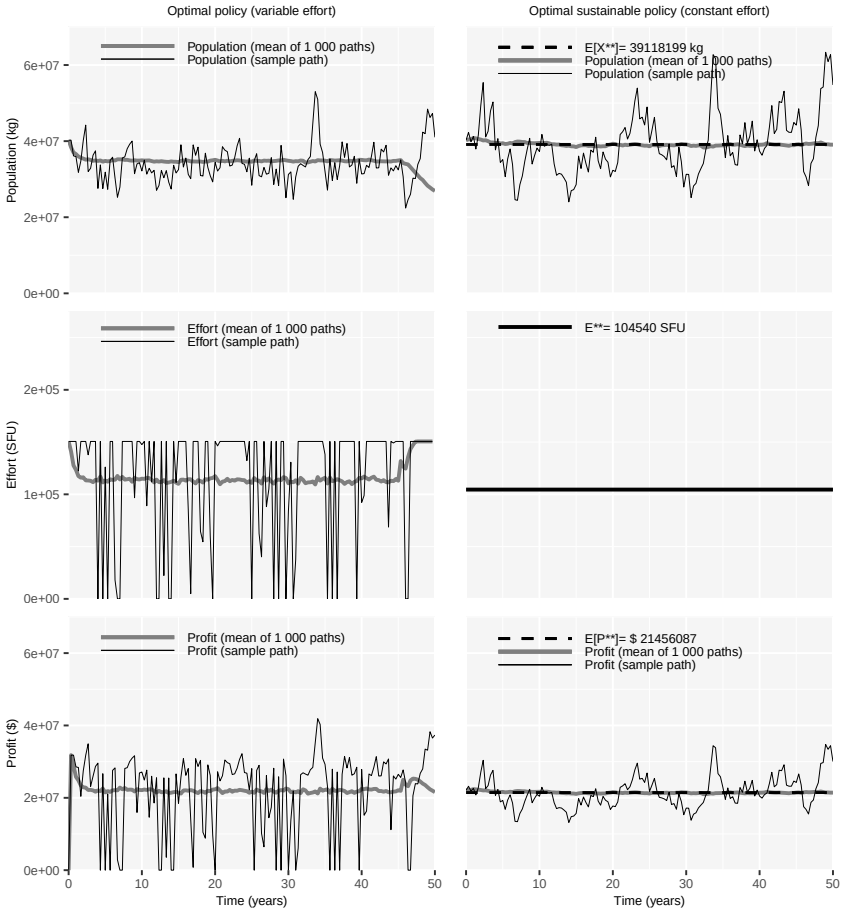

Figure 6: Scenario S_6: mean and randomly chosen sample path for the population, the effort and the profit per unit time. The optimal variable effort policy is on the left side and the optimal constant effort sustainable policy is on the right side.

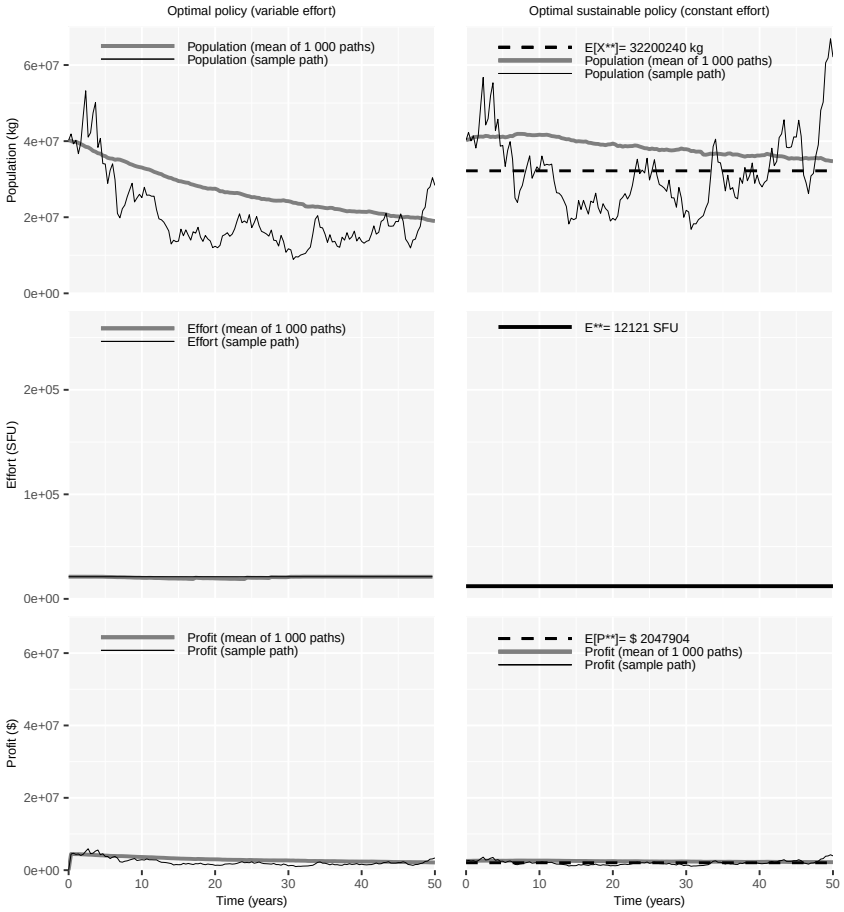

Figure 7: Scenario S_7: mean and randomly chosen sample path for the population, the effort and the profit per unit time. The optimal variable effort policy is on the left side and the optimal constant effort sustainable policy is on the right side.

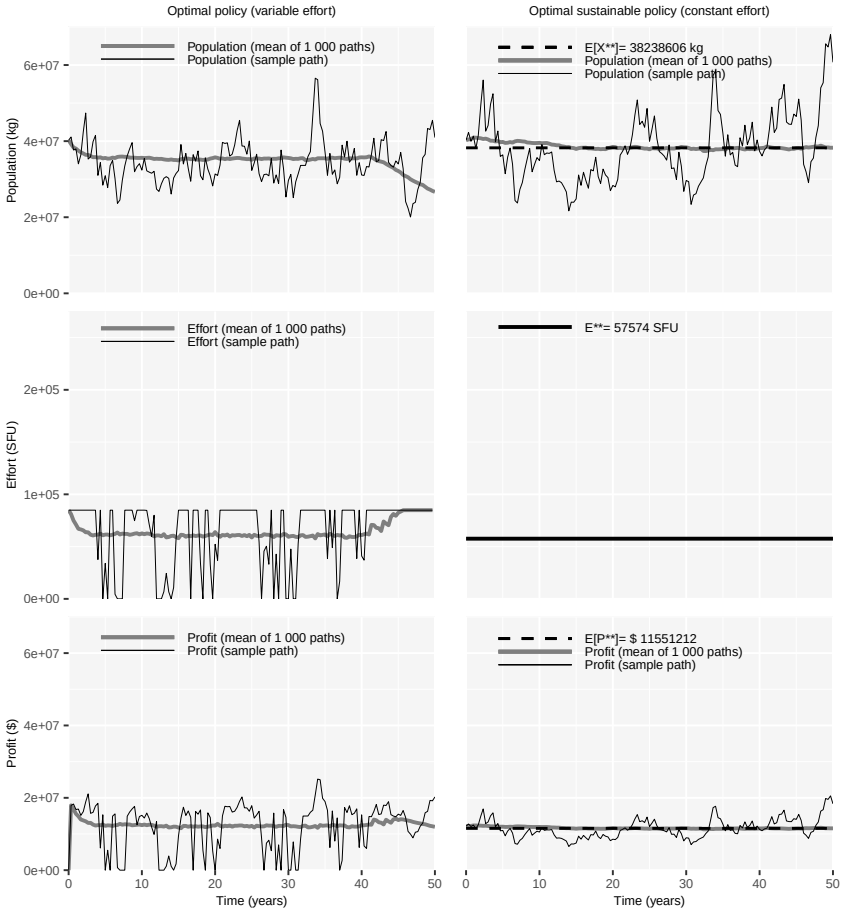

Figure 8: Scenario S_8: mean and randomly chosen sample path for the population, the effort and the profit per unit time. The optimal variable effort policy is on the left side and the optimal constant effort sustainable policy is on the right side.

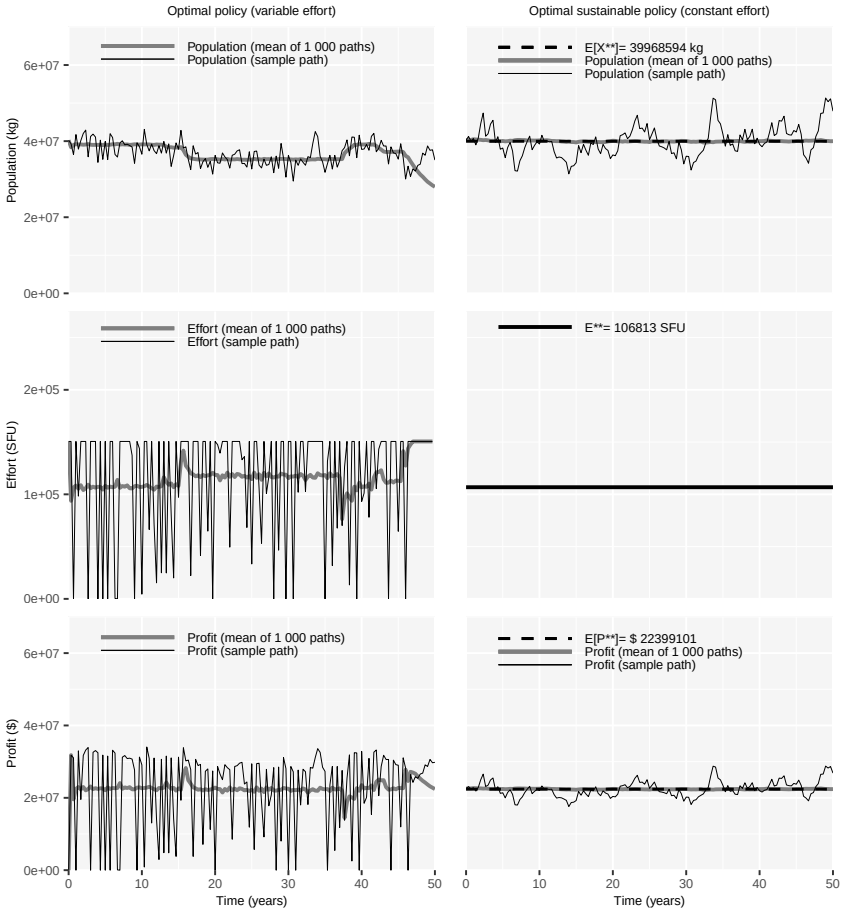

Figure 9: Scenario S_9: mean and randomly chosen sample path for the population, the effort and the profit per unit time. The optimal variable effort policy is on the left side and the optimal constant effort sustainable policy is on the right side.

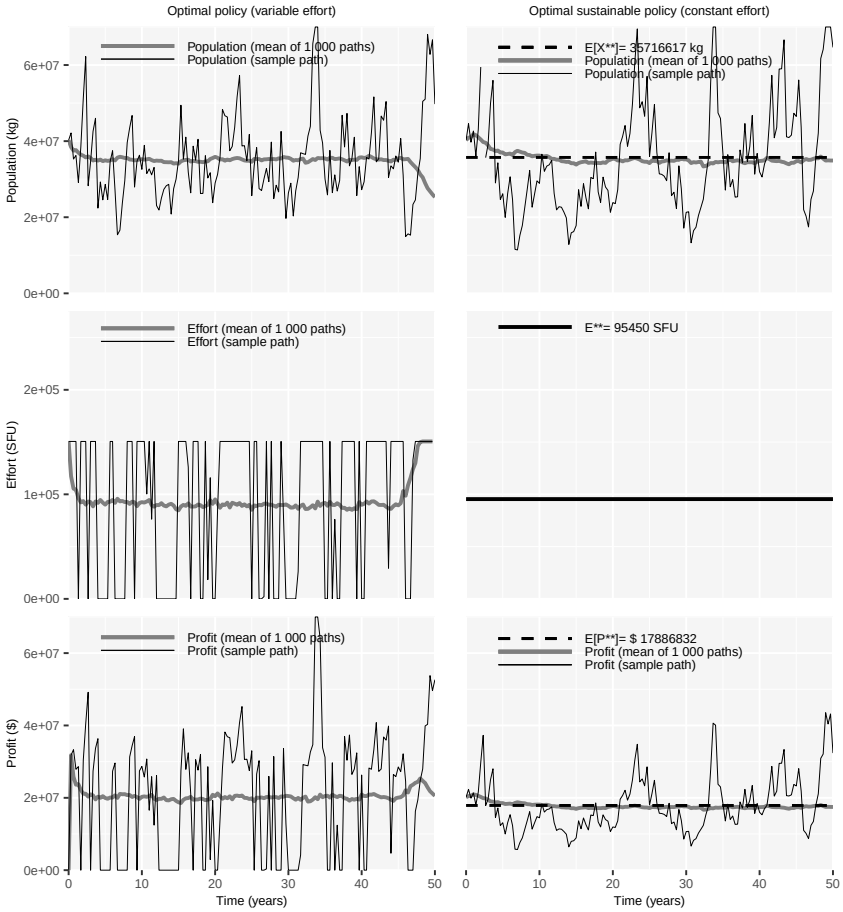

Figure 10: Scenario S_{10}: mean and randomly chosen sample path for the population, the effort and the profit per unit time. The optimal variable effort policy is on the left side and the optimal constant effort sustainable policy is on the right side.

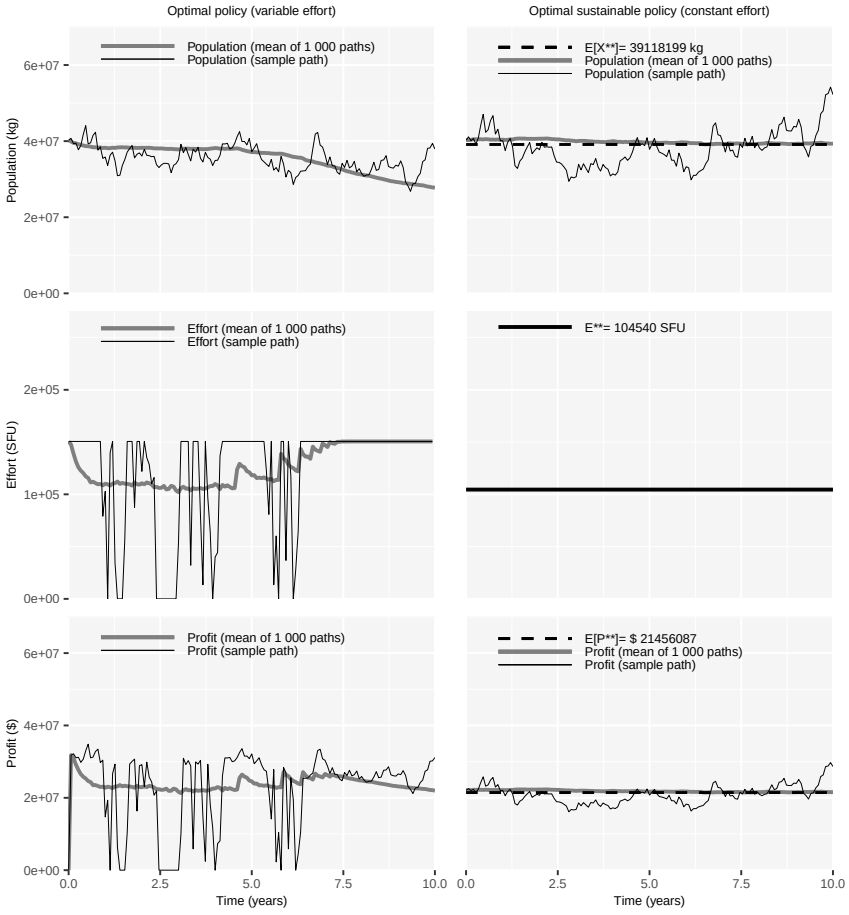

Figure 11: Scenario S_{11}: mean and randomly chosen sample path for the population, the effort and the profit per unit time. The optimal variable effort policy is on the left side and the optimal constant effort sustainable policy is on the right side.

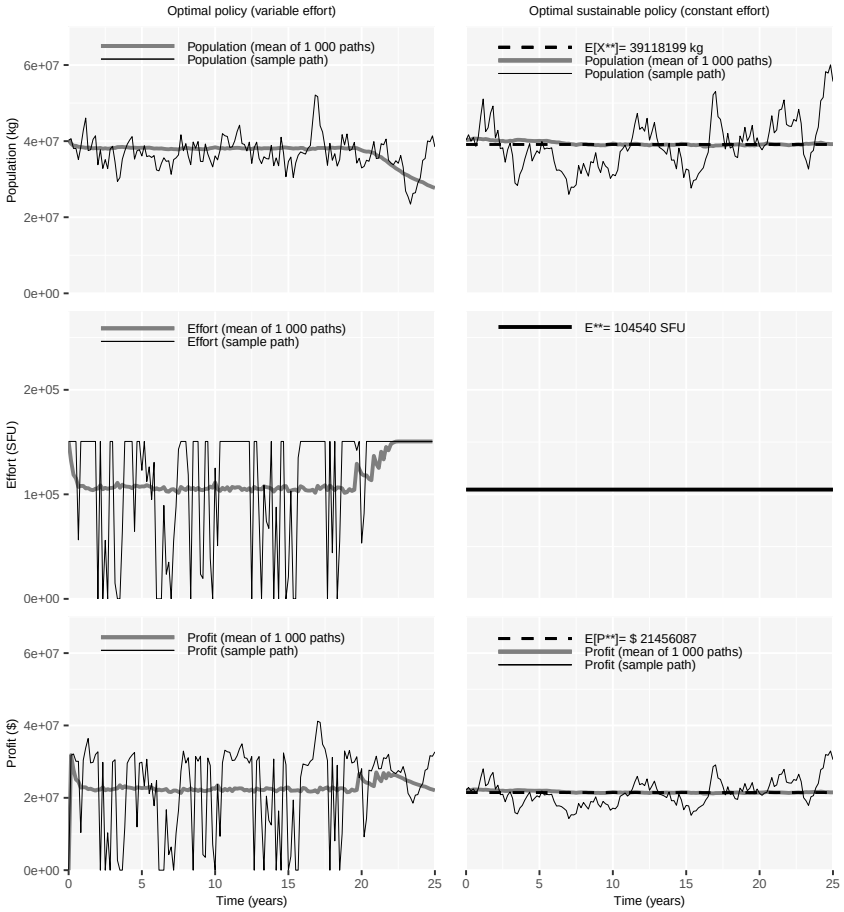

Figure 12: Scenario S_{12}: mean and randomly chosen sample path for the population, the effort and the profit per unit time. The optimal variable effort policy is on the left side and the optimal constant effort sustainable policy is on the right side.

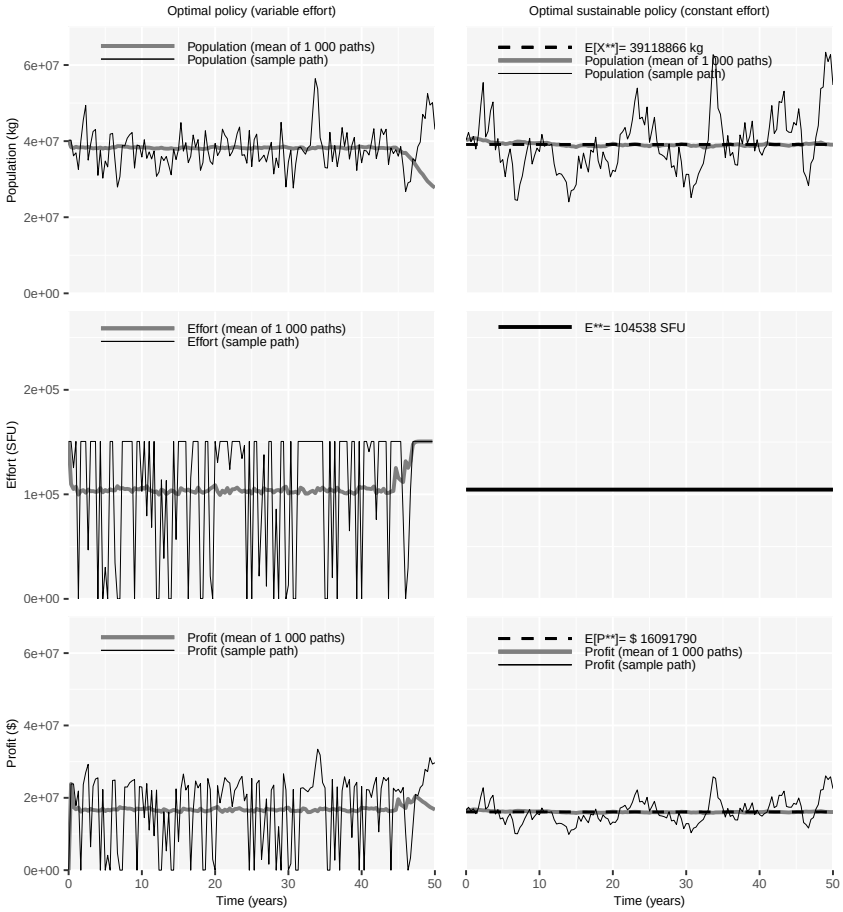

Figure 13: Scenario S_{13}: mean and randomly chosen sample path for the population, the effort and the profit per unit time. The optimal variable effort policy is on the left side and the optimal constant effort sustainable policy is on the right side.

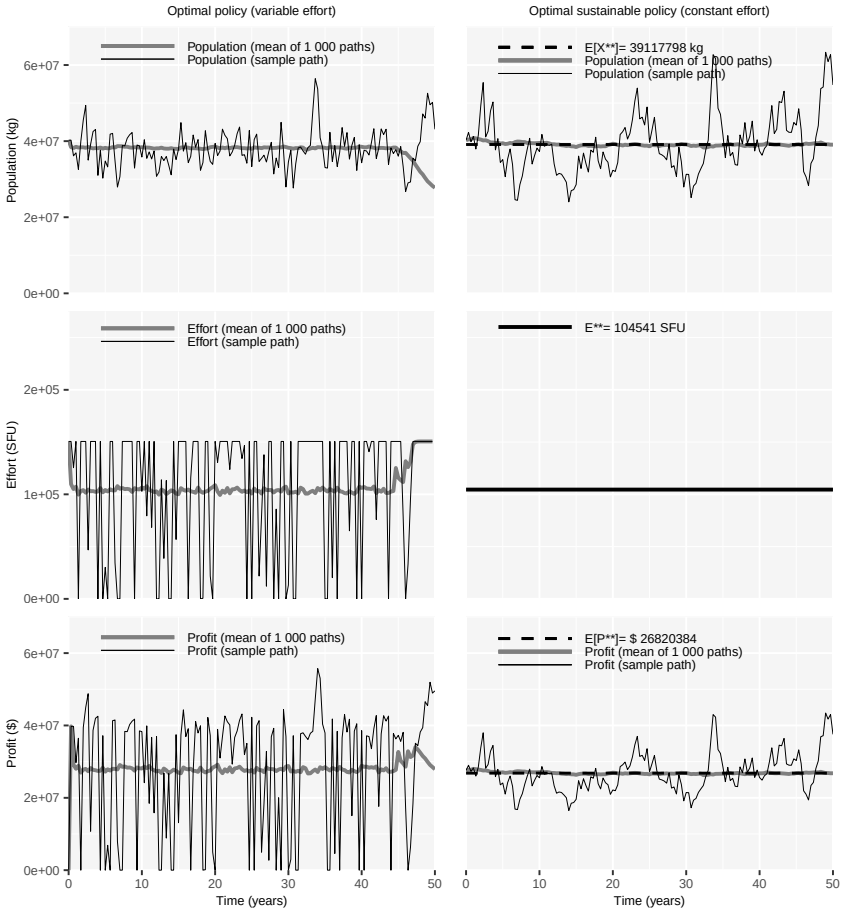

Figure 14: Scenario S_{14}: mean and randomly chosen sample path for the population, the effort and the profit per unit time. The optimal variable effort policy is on the left side and the optimal constant effort sustainable policy is on the right side.

Note: Figures for the scenarios S_{15}, S_{16}, S_{17} and S_{18} are not shown since they are almost undistinguishable from the figure associated to the basic scenario S_0.

2 R code for computations and graphics

```
# Main code

rm(list = ls(all = TRUE))

set.seed(123456789)

setwd("~/Desktop/Rcode")

g<-basename(getwd())

library(matrixcalc)

library(signal)

library(gridExtra)

# parmeters for scenario S0

# (for others, make appropriate changes)

T <- 50

r <- 0.71

K <- 80.5*10^6

q <- 3.30*10^(-6)

Emin <- 0

Emax <- 0.7*(r/q)

xmin <- 0

xmax <- 2*K

n <- 150

m <- 75

(deltat <- T/n)

(deltax <- (xmax - xmin)/m)

x <- seq(xmin, xmax, by = deltax)

t <- seq(0, T, by = deltat)

p <- 1.59

cone <- 96*10^(-6)
```

```
ctwo <- 0.10*10^(-6)
d <- 0.05
sigma <- 0.2
xone <- K/2
path <- 1000

J <- matrix(0, nrow = m + 1, ncol = n + 1)
E <- matrix(0, nrow = m + 1, ncol = n + 1)
A <- matrix(0, nrow = m, ncol = m)
B <- matrix(0, nrow = m, ncol = m)

# j for instants, i for states

for (j in 2:(n + 1)) {
for (i in 2:m) {
E[i, j - 1] <- (p - (J[i + 1, j - 1] - J[i - 1, j - 1])/(2 * deltax)) *
(q * x[i]/(2 * ctwo)) - cone/(2 * ctwo)
ifelse(E[i, j - 1] < Emin, E[i, j - 1] <- Emin, E[i, j - 1])
ifelse(E[i, j - 1] > Emax, E[i, j - 1] <- Emax, E[i, j - 1])
}

E[m + 1, j - 1] <- (p - (3 * J[m + 1, j - 1] - 4 * J[m, j - 1] + J[m -
1, j - 1])/(2 * deltax)) * (q * x[m + 1]/(2 * ctwo)) - cone/(2 *
ctwo)
ifelse(E[m + 1, j - 1] < Emin, E[m + 1, j - 1] <- Emin, E[m + 1, j -
1])
ifelse(E[m + 1, j - 1] > Emax, E[m + 1, j - 1] <- Emax, E[m + 1, j -
```

```
1])
# Solving HJB equation
source("func.aux1.R")
source("func.aux2.R")
source("func.aux3.R")
c1 <- rep(0, m + 1)
c2 <- rep(0, m + 1)
c3 <- rep(0, m + 1)
for (i in 2:(m + 1)) {
c1[i] <- constant1(r, K, q, x[i], E[i, j - 1])
c2[i] <- constant2(deltat, deltax, sigma, x[i])
c3[i] <- constant3(deltat, p, q, cone, ctwo, x[i], E[i, j - 1])
}
A[1, 1] <- 1 + d * deltat/2 + 0.5 * c2[2]
A[1, 2] <- -deltat * c1[2]/(4 * deltax) - 0.25 * c2[2]
A[m, m - 3] <- 0.25 * c2[m + 1]
A[m, m - 2] <- -deltat * c1[m + 1]/(4 * deltax) - 0.25 * c2[m + 1]
A[m, m - 1] <- deltat * c1[m + 1]/deltax + 1.25 * c2[m + 1]
A[m, m] <- 1 + d * deltat/2 - 3 * deltat * c1[m + 1]/(4 * deltax) -
0.5 * c2[m + 1]
firstcolumn <- 1
for (row in 2:(m - 1)) {
A[row, firstcolumn] <- deltat * c1[row + 1]/(4 * deltax) - 0.25 *
c2[row + 1]
A[row, firstcolumn + 1] <- 1 + d * deltat/2 + 0.5 * c2[row + 1]
A[row, firstcolumn + 2] <- -deltat * c1[row + 1]/(4 * deltax) -
0.25 * c2[row + 1]
```

```
firstcolumn <- firstcolumn + 1
}
B[1, 1] <- 1 - d * deltat/2 - 0.5 * c2[2]
B[1, 2] <- deltat * c1[2]/(4 * deltax) + 0.25 * c2[2]
B[m, m - 3] <- -0.25 * c2[m + 1]
B[m, m - 2] <- deltat * c1[m + 1]/(4 * deltax) + 0.25 * c2[m + 1]
B[m, m - 1] <- -deltat * c1[m + 1]/deltax - 0.5 * c2[m + 1]
B[m, m] <- 1 - d * deltat/2 + 3 * deltat * c1[m + 1]/(4 * deltax) +
0.25 * c2[m + 1]
firstcolumn <- 1
for (row in 2:(m - 1)) {
B[row, firstcolumn] <- -deltat * c1[row + 1]/(4 * deltax) + 0.25 *
c2[row + 1]
B[row, firstcolumn + 1] <- 1 - d * deltat/2 - 0.5 * c2[row + 1]
B[row, firstcolumn + 2] <- deltat * c1[row + 1]/(4 * deltax) +
0.25 * c2[row + 1]
firstcolumn <- firstcolumn + 1
}
C <- B %*% J[2:(m + 1), (j - 1)]
D <- c3[2:(m + 1)]
F <- C + D
lu.dec <- lu.decomposition(A)
L <- lu.dec$L
U <- lu.dec$U
temp.mat <- solve(U) %*% solve(L) %*% F
for (count in 2:(m + 1)) {
J[count, j] <- temp.mat[count - 1]
```

```
}
}
for (i in 2:m) {
E[i, n + 1] <- (p - (J[i + 1, n + 1] - J[i - 1, n + 1])/(2 * deltax)) *
(q * x[i]/(2 * ctwo)) - cone/(2 * ctwo)
ifelse(E[i, n + 1] < Emin, E[i, n + 1] <- 0, E[i, n + 1])
ifelse(E[i, n + 1] > Emax, E[i, n + 1] <- Emax, E[i, n + 1])
}
E[m + 1, n + 1] <- (p - (3 * J[m + 1, n + 1] - 4 * J[m, n + 1] + J[m -
1, n + 1])/(2 * deltax)) * (q * x[m + 1]/(2 * ctwo)) - cone/(2 * ctwo)
ifelse(E[m + 1, n + 1] < Emin, E[m + 1, n + 1] <- 0, E[m + 1, n + 1])
ifelse(E[m + 1, n + 1] > Emax, E[m + 1, n + 1] <- Emax, E[m + 1, n + 1])

write.table(round(J, digits = 0), "J.csv", sep = ";", dec = ",", row.names = FALSE)
write.table(round(E, digits = 0), "E.csv", sep = ";", dec = ",", row.names = FALSE)

# Interpolation

population <- matrix(xone, nrow = path, ncol = n + 1)
Estar <- matrix(0, nrow = path, ncol = n + 1)
Jstar <- matrix(0, nrow = path, ncol = n + 1)
# Wiener process
mat.estoc <- matrix(rnorm(path * (n + 1)), nrow = path, ncol = n + 1)
dW <- (sqrt(deltat)) * mat.estoc
W <- t(apply(dW, 2, sum))
for (i in 1:path) {
for (b in 1:(n)) {
```

```
if (population[i, b] < xmax & population[i, b] > xmin) {
Estar[i, b] <- interp1(x, E[, (n + 1) - (b - 1)], population[i,
b], method = "linear", extrap = FALSE)
Jstar[i, b] <- interp1(x, J[, (n + 1) - (b - 1)], population[i,
b], method = "linear", extrap = FALSE)
} else {
Estar[i, b] <- interp1(x, E[, (n + 1) - (b - 1)], population[i,
b], method = "linear", extrap = TRUE)
Jstar[i, b] <- interp1(x, J[, (n + 1) - (b - 1)], population[i,
b], method = "linear", extrap = TRUE)
}
if (Estar[i, b] < 0) {
Estar[i, b] <- 0
}
if (Estar[i, b] > Emax) {
Estar[i, b] <- Emax
}
# Euler scheme
population[i, b + 1] <- population[i, b] + (r * population[i, b] *
(1 - (population[i, b])/K) - q * Estar[i, b] * population[i,
b]) * deltat + sigma * population[i, b] * dW[i, b]
}
}

Estarmean <- apply(Estar, 2, mean)
Jstarmean <- apply(Jstar, 2, mean)
populationmean <- apply(population, 2, mean)
```

```
dWmean <- apply(dW, 2, mean)

#Sustainable policy
Estarstar <- (p * q * K * (r - 0.5 * sigma^2) - cone * r)/(2 * p * (q^2) *
K + 2 * r * ctwo)
Estarstar
EXstarstar <- K * (1 - (p * (q^2) * K * (r - 0.5 * sigma^2) - cone * r *
q)/(2 * r * p * (q^2) * K + 2 * (r^2) * ctwo) - (sigma^2)/(2 * r))
EXstarstar
Pistarstar <- ((r - 0.5 * sigma^2) * p * q * K - cone * r)^2/(4 * r * (p *
(q^2) * K + r * ctwo))
Pistarstar

write.table(round(populationmean, digits = 0), "Estar.csv", sep = ";",
dec = ",", row.names = FALSE)
write.table(round(Estar, digits = 0), "Estar.csv", sep = ";", dec = ",",
row.names = FALSE)
write.table(round(Jstar, digits = 0), "Jstar.csv", sep = ";", dec = ",",
row.names = FALSE)
write.table(round(Estarmean, digits = 0), "Estarmean.csv", sep = ";", dec = ",",
row.names = FALSE)
write.table(round(Jstarmean, digits = 0), "Jstarmean.csv", sep = ";", dec = ",",
row.names = FALSE)

##### Comparisons #####

#Auxiliar variable
```

```
expdt <- deltat * matrix(rep((exp(-d * t) +
exp(-d * (t + deltat)))/2, path), nrow = path, ncol = n + 1, byrow = T)

############
###  1  ###
############
(PistarT <- mean(Jstar[, 1]))

#############
###  1A  ###
#############
expdtLEstar <- expdt * (p * q * population - cone - ctwo * Estar) * Estar
intexpdtLEstar <- apply(expdtLEstar[,1:n], 1, sum)
(PistarT2 <- mean(intexpdtLEstar))

############
###  2  ###
############
LEstar <- (p * q * population - cone - ctwo * Estar) * Estar
LEstardt <- LEstar * deltat
intLEstardt <- apply(LEstardt[,1:n], 1, sum)
(VstarT <- mean(intLEstardt))

############
###  3  ###
############
ifelse(d==0,int0T <- T,int0T <- (1 - exp(-d * T))/d)
```

```
(PistarT2/int0T)

############
###  4  ###
############
(VstarT/T)

############
###  5  ###
############
expdt <- deltat * matrix(rep(exp(-d * t),path), nrow = path, ncol = n + 1, byrow = T)
simX <- matrix(0, nrow = path, ncol = n + 1)
simX[, 1] <- rep(xone, path)
for (i in 1:path) {
for (j in 1:(n)) {
simX[i, j + 1] <- simX[i, j] + (r * simX[i, j] * (1 - (simX[i,
j])/K) - q * Estarstar * simX[i, j]) * deltat + sigma * simX[i,
j] * dW[i, j]
}
}
expdtLEstarstar <- expdt * (p * q * simX - cone - ctwo * Estarstar) * Estarstar
intexpdtLEstarstar <- apply(expdtLEstarstar, 1, sum)
(PistarstarT <- mean(intexpdtLEstarstar))

############
###  6  ###
############
```

```
LEstarstar <- (p * q * simX - cone - ctwo * Estarstar) * Estarstar
LEstarstardt <- LEstarstar * deltat
intLEstarstardt <- apply(LEstarstardt[,1:n], 1, sum)
(VstarstarT <- mean(intLEstarstardt))

############
###  7  ###
############
ifelse(d==0,int0T <- T,int0T <- (1 - exp(-d * T))/d)
(PistarstarT/int0T)

############
###  8  ###
############
(VstarstarT/T)

####################################
###  Print values to a pdf file  ###
####################################
pdf(paste("Comparisons",g,'.pdf',sep=''), height=5, width=5)
V.old <- c(PistarT,0,0,0,0)
V.star <- c(PistarT2,sd(intexpdtLEstar),PistarstarT,sd(intexpdtLEstarstar),
(PistarstarT-PistarT2)/PistarT2)
V.star.u <- c(VstarT,sd(intLEstardt),VstarstarT,sd(intLEstarstardt),
(VstarstarT-VstarT)/VstarT)
P.star <- c(PistarT2/int0T,sd(intexpdtLEstar)/int0T,PistarstarT/int0T,
sd(intexpdtLEstarstar)/int0T,((PistarstarT/int0T)-(PistarT2/int0T))/(PistarT2/int0T))
```

```
P.star.u <- c(VstarT/T,sd(intLEstardt)/T,VstarstarT/T,sd(intLEstarstardt)/T,
((VstarstarT/T)-(VstarT/T))/(VstarT/T))
data.mat <- matrix(c(V.old,V.star,V.star.u,P.star,P.star.u),5,5,byrow=TRUE)
data.mat[,1:4] <- round(10^(-6)*data.mat[,1:4],3)
data.mat[,5] <- round(100*data.mat[,5],1)
colnames(data.mat) <- c('Opt.','sd','Opt. Sust.','sd','%')
rownames(data.mat) <- c('V.old','V.star','V.star.u','P.star','P.star.u')
data.mat
grid.table(data.mat)
dev.off()
##### End comparisons ######
# Profit plot
matLopt <- matrix(0, nrow = path, ncol = n + 1)
matLopt[, 1] <- rep(0, path)
for (i in 1:path) {
for (j in 1:n) {
matLopt[i, j + 1] <- (p * q * population[i, j] - cone - ctwo *
Estar[i, j]) * Estar[i, j]
}
}
Lopt <- apply(matLopt, 2, mean)
Lsust <- (p * q * simX - cone - ctwo * Estarstar) * Estarstar
Lsust <- apply(Lsust, 2, mean)

matLEstarstar <- (p * q * simX - cone - ctwo * Estarstar) * Estarstar

# random path
```

```
(num1<-floor(runif(1)*path)+1)

#graphics
library(ggplot2)
source("func.aux4.R")

df11<-data.frame(Time=t,Population=populationmean)
df12<-data.frame(Time=t,Population=population[num1,])
plot1<-ggplot(df11, aes(Time,Population))+geom_line(size=1,aes(color=
'Population (mean of 1 000 paths)'))+
scale_y_continuous(limits = c(0,7e+07), expand = c(0, 0)) +
scale_x_continuous(limits = c(0,T), expand = c(0, 0)) +
geom_line(data=df12,size=0.25,aes(color=
'Population (sample path)'))+labs(y='Population (kg)',color='Legend')+
theme(legend.justification=c(0,1.10), legend.position=c(0,1.10),
legend.background= element_rect(fill=alpha('white', 0)))+
theme(legend.key.size =  unit(0.15, "in"))+
theme(legend.text = element_text(size=7))+
theme(legend.title=element_blank())+
ggtitle("Optimal policy (variable effort)")+
theme(plot.title = element_text(size = 7,hjust=0.5))+
theme(axis.text=element_text(size=7),
axis.title=element_text(size=7))+
theme(axis.title.x=element_blank(),
axis.text.x=element_blank(),
axis.ticks.x=element_blank())+
theme(plot.margin= unit(c(.5, .5, 0, .5), "lines"))+
```

```
theme(legend.key = element_blank())+
theme(panel.background = element_rect(fill = "grey96",
colour = "grey96",
size = 0.5, linetype = "solid"))+
guides(colour = guide_legend(keywidth = 3.5,
keyheight = .5,override.aes = list(linetype=c('solid','solid'),size=c(1,0.25))))+
scale_color_manual(values=c("grey50", "black"))

df21<-data.frame(Time=t[1:n],Effort=Estarmean[1:n])
df22<-data.frame(Time=t[1:n],Effort=Estar[num1, 1:n])
plot2<-ggplot(df21, aes(Time,Effort))+geom_line(size=1,
aes(color='Effort (mean of 1 000 paths)'))+
scale_y_continuous(limits = c(0,2.75e+05), expand = c(0, 0)) +
scale_x_continuous(limits = c(0,T), expand = c(0, 0)) +
geom_line(data=df22,size=0.25,
aes(color='Effort (sample path)'))+labs(y='Effort (SFU)',x='Time (years)',color='Legend')+
theme(legend.justification=c(0,1.10),
legend.position=c(0,1.10),
legend.background = element_rect(fill=alpha('white', 0)))+
theme(legend.key.size =  unit(0.15, "in"))+
theme(legend.text = element_text(size=7))+
theme(legend.title=element_blank())+
theme(axis.text=element_text(size=7),
axis.title=element_text(size=7))+
theme(axis.title.x=element_blank(),
axis.text.x=element_blank(),
axis.ticks.x=element_blank())+
```

```
theme(plot.margin= unit(c(.5, .5, 0, .5), "lines"))+
theme(legend.key = element_blank())+
theme(panel.background = element_rect(fill = "grey96",
colour = "grey96",
size = 0.5, linetype = "solid"))+
guides(colour = guide_legend(keywidth = 3.5,
keyheight = .5,override.aes = list(linetype=c('solid','solid'),size=c(1,0.25))))+
scale_color_manual(values=c("grey50", "black"))



df31<-data.frame(Time=t,Population=apply(simX, 2, mean))
df32<-data.frame(Time=t,Population=simX[num1,])
df33<-data.frame(Time=t,Population=rep(EXstarstar, n + 1))
plot3<-ggplot(df31, aes(Time,Population))+geom_line(size=1,
aes(color='Population (mean of 1 000 paths)'))+
scale_y_continuous(limits = c(0,7e+07), expand = c(0, 0)) +
scale_x_continuous(limits = c(0,T), expand = c(0, 0)) +
geom_line(data=df32,size=0.25,
aes(color='Population (sample path)'))+
labs(y='Population (kg)',color='Legend')+
geom_line(linetype = "dashed",data=df33,size=.75,aes(color=paste("E[X**]=",
round(EXstarstar, 0),'kg')))+
scale_color_manual(values=c("black", "grey50", "black"))+
theme(legend.justification=c(0,1.10),
legend.position=c(0,1.10),legend.background =
element_rect(fill=alpha('white', 0)))+
theme(legend.text = element_text(size=7))+
```

```
theme(legend.title=element_blank())+
ggtitle("Optimal sustainable policy (constant effort)")+
theme(plot.title = element_text(size = 7,hjust=0.5))+
theme(axis.text=element_text(size=7),
axis.title=element_text(size=7))+
theme(axis.title.y=element_blank(),
axis.text.y=element_blank(),
axis.ticks.y=element_blank())+
theme(axis.title.x=element_blank(),
axis.text.x=element_blank(),
axis.ticks.x=element_blank())+
theme(plot.margin= unit(c(.5, 2.8, 0, .5), "lines"))+
theme(legend.key = element_blank())+
theme(
panel.background = element_rect(fill = "grey96",
colour = "grey96",
size = 0.5, linetype = "solid"))+
guides(colour = guide_legend(keywidth = 3.5,
keyheight = .5,override.aes = list(linetype=
c('dashed','solid','solid'),size=c(.75,1,0.25))))

df41<-data.frame(Time=t,Effort=rep(Estarstar, n + 1))
plot4<-ggplot(df41, aes(Time,Effort))+
geom_line(size=1,aes(color=c(paste("E**=",
round(Estarstar, 0),'SFU'))))+
scale_y_continuous(limits = c(0,2.75e+05), expand = c(0, 0)) +
scale_x_continuous(limits = c(0,T), expand = c(0, 0)) +
```

```
labs(y='Effort (SFU)', x='Time (years)',color='Legend')+
theme(legend.justification=c(0,1.10),
legend.position=c(0,1.10), legend.background =
element_rect(fill=alpha('white', 0)))+
theme(legend.key.size =  unit(0.15, "in"))+
theme(legend.text = element_text(size=7))+
theme(legend.title=element_blank())+
theme(axis.text=element_text(size=7),
axis.title=element_text(size=7))+
theme(axis.title.x=element_blank(),
axis.text.x=element_blank(),
axis.ticks.x=element_blank())+
theme(axis.title.y=element_blank(),
axis.text.y=element_blank(),
axis.ticks.y=element_blank())+
guides(colour = guide_legend(keywidth = 3.5,
keyheight = .5,override.aes =
list(linetype=c('solid'),size=1)))+
theme(plot.margin= unit(c(.5, 2.8, 0, .5), "lines"))+
theme(legend.key = element_blank())+
theme(panel.background = element_rect(fill = "grey96",
colour = "grey96",
size = 0.5, linetype = "solid"))+
scale_color_manual(values=c("black"))


df51<-data.frame(Time=t,Profit=Lopt)
```

```
df52<-data.frame(Time=t,Profit=matLopt[num1,])
plot5<-ggplot(df51, aes(Time,Profit))+
geom_line(size=1,aes(color='Profit (mean of 1 000 paths)'))+
scale_y_continuous(limits = c(0,7e+07), expand = c(0, 0)) +
scale_x_continuous(limits = c(0,T), expand = c(0, 0)) +
geom_line(data=df52,size=0.25,
aes(color='Profit (sample path)'))+
labs(y='Profit ($)',x='Time (years)',color='Legend')+
theme(legend.justification=c(0,1.10),
legend.position=c(0,1.10),
legend.background = element_rect(fill=alpha('white', 0)))+
theme(legend.key.size =  unit(0.15, "in"))+
theme(legend.text = element_text(size=7))+
theme(legend.title=element_blank())+
theme(axis.text=element_text(size=7),
axis.title=element_text(size=7))+
theme(plot.margin= unit(c(.5, .5, 0, .5), "lines"))+
theme(legend.key = element_blank())+
theme(panel.background = element_rect(fill = "grey96",
colour = "grey96",
size = 0.5, linetype = "solid"))+
guides(colour = guide_legend(keywidth = 3.5,
keyheight = .5,override.aes =
list(linetype=c('solid','solid'),size=c(1,0.25))))+
scale_color_manual(values=c("grey50", "black"))

df61<-data.frame(Time=t,Profit=Lsust)
```

```
df62<-data.frame(Time=t,Profit=matLEstarstar[num1,])
df63<-data.frame(Time=t,Profit=rep(Pistarstar, n + 1))
plot6<-ggplot(df61, aes(Time,Profit))+
geom_line(size=1,aes(color='Profit (mean of 1 000 paths)'))+
scale_y_continuous(limits = c(0,7e+07), expand = c(0, 0)) +
scale_x_continuous(limits = c(0,T), expand = c(0, 0)) +
geom_line(data=df62,size=0.25,
aes(color='Profit (sample path)'))+
labs(y='Profit ($)',color='Legend', x='Time (years)')+
geom_line(linetype = "dashed",data=df63,
size=.75,aes(color=paste("E[P**]=",'$', round(Pistarstar, 0))))+
scale_color_manual(values=c("black", "grey50", "black"))+
theme(legend.justification=c(0,1.10),
legend.position=c(0,1.10),legend.background =
element_rect(fill=alpha('white', 0)))+
theme(legend.text = element_text(size=7))+
theme(legend.title=element_blank())+
theme(plot.title = element_text(size = 7))+
theme(axis.text=element_text(size=7),
axis.title=element_text(size=7))+
theme(axis.title.y=element_blank(),
axis.text.y=element_blank(),
axis.ticks.y=element_blank())+
theme(plot.margin= unit(c(.5, 2.8, 0, .5), "lines"))+
theme(legend.key = element_blank())+
theme(panel.background = element_rect(fill = "grey96",
colour = "grey96",
```

```
size = 0.5, linetype = "solid"))+
guides(colour = guide_legend(keywidth = 3.5,
keyheight = .5,override.aes =
list(linetype=c('dashed','solid','solid'),
size=c(.75,1,.25))))
g1 <- ggplotGrob(plot1)
g2 <- ggplotGrob(plot2)
g3 <- ggplotGrob(plot3)
g4 <- ggplotGrob(plot4)
g5 <- ggplotGrob(plot5)
g6 <- ggplotGrob(plot6)
ggsave(paste('britesbraumann',g,'.pdf',sep=''),
grid.arrange(g1, g3, g2, g4, g5, g6, ncol=2),width=7,height=7)
save.image(paste(g,'.RData',sep=''))
###End
# Aux. function 1
constant1<-function(r,K,q,X,E){
return(r*X*(1-X/K)-q*E*X)
}
# Aux. function 2
constant2<-function(deltat,deltax,sigma,X){
return(deltat*(sigma^2)*(X^2)/(deltax^2))
}
# Aux. function 3
constant3<-function(deltat,p,q,c1,c2,X,E){
return(deltat*E*(p*q*X-cone-ctwo*E))
}
```

```
# Aux. function 4
multiplot <- function(..., plotlist=NULL, file, cols=1, layout=NULL) {
library(grid)
# Make a list from the ... arguments and plotlist
plots <- c(list(...), plotlist)
numPlots = length(plots)
# If layout is NULL, then use 'cols' to determine layout
if (is.null(layout)) {
# Make the panel
# ncol: Number of columns of plots
# nrow: Number of rows needed, calculated from # of cols
layout <- matrix(seq(1, cols * ceiling(numPlots/cols)),
ncol = cols, nrow = ceiling(numPlots/cols))
}
if (numPlots==1) {
print(plots[[1]])
} else {
# Set up the page
grid.newpage()
pushViewport(viewport(layout = grid.layout(nrow(layout), ncol(layout))))
# Make each plot, in the correct location
for (i in 1:numPlots) {
# Get the i,j matrix positions of the regions that contain this subplot
matchidx <- as.data.frame(which(layout == i, arr.ind = TRUE))
print(plots[[i]], vp = viewport(layout.pos.row = matchidx$row,
layout.pos.col = matchidx$col))}}}
```