POLITECNICO DI MILANO

SIMULATION, LEARNING AND CONTROL FOR BUSINESS APPLICATIONS

ANALYSIS OF THE FISHERY MODEL

Professor: Gabriele Ciaramella

Author:
Enver Eren, 10947639

Contents

1.Introduction and description of the Fishery Resource Model

1.1 Overview of the model

The Fishery Resource Model is an example of a broader problem, the one regarding the usage of natural resources that in order to be used by humans in the long run need to reproduce themselves. In particular, the rate of consumption of the resource needs to be smaller than the rate at which the resource is able to reproduce itself, to avoid scarcity of such resource. Currently many examples of these processes exist, from the different resources of energy to the level of water of rivers and lake, and in this report the fishing activity will be discussed and thoroughly analyzed. The fishing problem considers mainly the natural growth function of fish, the fishing effort and with known values of unitary revenues and costs tries to maximize the profits.

1.2 Variables, Constraints, and objective function

The meaningful items with the following notation for the problem (p.312 Sethi) are:

State variable:

x(t): the biomass of the fish population at time t expressed in tons

Control variable:

u(t): the rate of fishing effort at time t expressed in tons of potential fish caught

Parameters:

g(x): natural growth function of the fish

q: the catchability coefficient (efficiency of the fishing process)

p: the unit price of landed fish in €

c: the unit cost of fishing effort in €

ρ: the discount rate

Considering the control function, U denotes the maximum fishing effort such that: 0 < u < U.

The state equation, due to Gordon and Schaefer, is

$$\dot{x}(t) = g\big(x(t)\big) - qu(t)x(t), \qquad x(0) = x_0$$

where qux is the catching rate assumed to be proportional to the rate of fishing effort and the biomass.

The profit rate is:

$$\pi(x,u) = \ pqux \ - \ cu = (pqx - c)\ u$$

Therefore, the objective function for the optimal control problem that attempts to maximize the profit in a given finite horizon T is:

$$\text{MAX} \quad \text{J} = \int_0^T e^{-\rho t}(pqx(t) - c)u(t)\ dt$$

Subject to:

$$\dot{x}(t) = g\big(x(t)\big) - qu(t)x(t), \qquad x(0) = x_0$$
$$0 \le u(t) \le U \qquad \forall t \in [0,T]$$

Inversely the objective function can be rewritten as a minimization problem:

$$\text{MIN J} = \int_0^T e^{-\rho t}\big(c - \ pqx(t)\big)u(t)\ dt$$

Subject to:

$$\dot{\text{x}}(\text{t}) = \text{g}\big(\text{x}(\text{t})\big) - \text{qu}(\text{t})\text{x}(\text{t}), \qquad \text{x}(0) = \text{x}_0$$
$$0 \le \text{u}(\text{t}) \le \text{U} \qquad \forall \text{t} \in [0,\text{T}]$$

To perform the analysis these are the choices for the values of the parameters and the function expressions described above, according to research and estimations.

ρ=0.1

The discount rate is a percentage used to actualize cash flows that will occur over time. Its value is normally around 10% therefore for this model the discount rate will be equal to 0.1.

x(t)= the tons of fish population at time t

To represent the biomass of fish population present different units of measure apply, tons will be used in this instance. In this example the external environment considered is a portion of the sea and the fish type is the Grass Carp.[1]

g(x)= 0,05x

The natural growth function of the fish depends on different constraints, including the type of fish and the condition of the natural environment in which they live. The trend considered in this model is linear, denoted by the function x with a coefficient which slows the growth. This may not be the best approximation, but it ensures that g is differentiable and concave.

Xo= 20 tons of fish

This represents the initial value of fish biomass in the sea portion considered.

Sea portion considered		
depth	10	m
width	75	m
length	75	m
VOLUME	56250	m^3
fish density	0,016	tons/m^3
X (max fish level)	900	tons
X0 (initial population)	20	tons

X= 900 tons

Considers the maximum level of fish that can be present: a research states that the average density of fish in 1 cube meter of seawater is roughly 16 kg. the value of X has been approximated using the assumption described in the table.

u(t): Control function

The rate of fishing effort is expressed in theoretical tons of fish caught, meaning that the potential amount of fish biomass that can be caught with the fishing method at time t is equal to u(t) tons. The fishing method could be one large net or multiple fishermen using fishing rods.

U= 900 tons

U is the upper bound of the control function.

q= 0.0001

The catchability coefficient is defined as the average proportion of fish biomass taken by each unit of fishing effort. This value ranges from 0 to 1 and it is volatile because it depends on many different factors such as period of the year, type of fish, effectiveness of the fishing methods and habits of the fish population (which might migrate further from the sea surface in certain parts of the day, see OFP.). Indeed, consider the catchability efficiency a constant might be inaccurate, but several studies on this matter have been conducted and it is difficult to establish a consistent relationship between how the change of the main factors truly affects q. Anyways its value is always very low, as in different case studies it is estimated to be between 0.001 and 0.00001 with cases in which the efficiency is even lower. For the project it will be estimated to be equal to 0.0001.

[1] It is the world's most common fish (weights around 25kg) and it has commercial catch of 5 million tons a year all over the world.

p= 3000 €/tons

The price is the value per tons of biomass caught, or the landed fish. It will be taken as reference the export price for carp of the United States, which is roughly 3 €/kg

c= 1€/Tons

This value considers the unitary cost of fishing effort, meaning how much it costs to try to catch 1 tons of biomass. Considering as the fishing method the netting, the average cost estimated is 1 €/Ton.

T= 50-time unit (year)

The time horizon is considered in time units because it could be days, months or years based on different assumptions.

2.Discretization of the Ordinary Differential Equation

The primary aim of this chapter is to acquire a set of difference equations that serve as an approximation to the provided differential equations. These difference equations solely focus on variations in function values and do not incorporate derivatives. Consequently, the solution to these difference equations corresponds to a discrete function defined at distinct points, deviating from the notion of a continuous function. To discretize the equations, the explicit Euler function, alternatively known as the forward Euler method, has been utilized. This numerical approach, known for its one-step methodology, is employed for the resolution of ordinary differential equations.

1) We discretize the time interval [t_0, T]:
   - $t_n = t_0 + n*h$, $h = T/N_h$, $n \in \{0, 1, 2, \ldots, N_h\}$
2) For every point t_n of this grid, we approximate the value of $x(t_n)$, $n \in \{0, 1, 2, \ldots, N_h\}$
   - $\dot{x}(t_n) = f(t_n, x(t_n))$, $n \in \{1, 2, \ldots, N_h\}$
   - $x(t_0) = x^0$
3) We denote the approximation by $x_n \approx x(t_n)$, $\forall n$
   - $\dot{x}(t_n) \approx \frac{x(t_{n+1}) - x(t_n)}{h}$
   - $f(t_n, x(t_n)) \approx f(t_n, x_n)$
   - $f(t_n, x_n) = \frac{x_{n+1} - x_n}{h}$
4) $x_{n+1} = x_n + h * f(t_n, x_n)$

By following the steps above, the discretized state equation for our problem is constructed below:

$$x_{i+1} = x_i + h(g(x_i) - qu_i x_i)$$

Also, we need to be sure about out the accuracy of discretization method. That's why we looked at the convergence of our method.

$$e_n = x(t_n) - x_n, \qquad n \in \{0, 1, 2, \ldots, Nh\}$$

A numerical model is convergent if: $\lim_{h \to 0} (\max_n |e_n|) = 0$

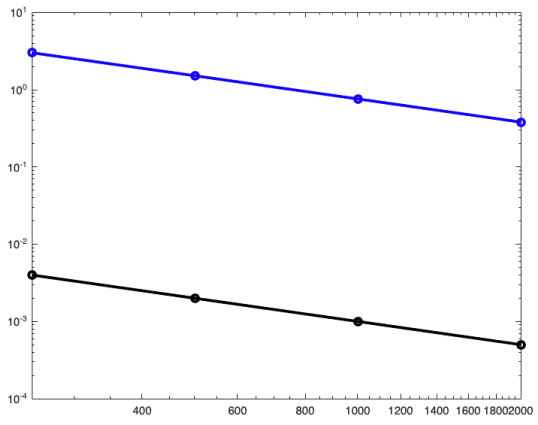
Moreover, it is convergent of order p if: $|e_n| = O(h^p)$, (i.e. p = 1 for explicit Euler)

So, for the different h's (step size) the error term must be linearly parallel to h's.

```
timestep = [1/5, 1/10, 1/20, 1/40];
ns = zeros(1,4);
errorexp = zeros(1,4);

for i=1:4
    [th, uh] = explicit_euler(u0, f, t0, tF, x0,  timestep(1,i));
    y_sol = y(th);
    errorexp(1,i) = max(abs(y_sol-uh));
    ns(1,i) = length(th);
end

loglog(ns,errorexp,"o-b",'LineWidth', 2.5)
hold on;
bound = ns.^(-1);
loglog(ns,bound,"o-k",'LineWidth', 2.5)
hold  off;
```



As you can see above, our method is convergent and order of convergence is 1.

3. Optimality systems (continuous and discrete)

Continuous Level:

Minimize J(x, u) = $\int_0^T e^{-\rho t}(c - pqx(t))u(t)\,dt + \frac{\beta}{2}\int_0^T (u(t))^2\,dt$

Subject to:

- $0 \le u(t) \le U \qquad \forall t \in [0, T]$
- $\dot{x}(t) = g(x(t)) - qu(t)x(t),\ x(0) = x_0$

Lagrangian function:

$L(x, u, \lambda) = J(x, u) + \int_{t=1}^{T} \lambda(t) f(x(t), u(t), t)dt + \lambda(t_0)(x(t_0) - x_0)$

Afterwards, the subsequent action involved the computation of the derivatives of the Lagrangian function concerning the state variables, control variables, and Lagrange multipliers. By equating these derivatives to zero, a collection of necessary and sufficient conditions is obtained, which must be met at the optimal solution. The conditions for each derivative can be summarized as follows:

1) $\dot{x}(t) = f(x(t), u(t), t) = g(x(t)) - qu(t)x(t),\ x(0) = x_0$ [derivative wrt λ]
2) $-\dot{\lambda}(t) = (g'(x(t)) - qu(t)) + (pqu(t)e^{-\rho t}),\ \lambda(T) = 0$ [derivative wrt x]
3) $\beta u(t) + qx(t) = 0$ [derivative wrt u]

Discrete Level:

Min $J_h = h * \sum_{i=1}^{N_h} e^{-\rho t_i}(c - pqx_i)u_i + \frac{\beta}{2} * h * \sum_{i=0}^{N_h - 1}(u_i)^2$ (1)

Subject to:

- $x_{i+1} = x_i + h(g(x_i) - qu_i x_i)$ (Explicit Euler) (2)
- $0 \le u_i \le U \qquad \forall i \in [0, N_h - 1]$

Derived from KKT equations:

- $\lambda_i = \lambda_{i+1} * (1 + h * (g_i' - qu_i)) + h * (pqu_i e^{-\rho t_i})$ [Adjoint equation] (3)
- $(\nabla \hat{J})_i = h * [\beta u_i + e^{-\rho t_i}(c - pqx_i) + qx_i\lambda_{i+1}]$ [Gradient equation] (4)

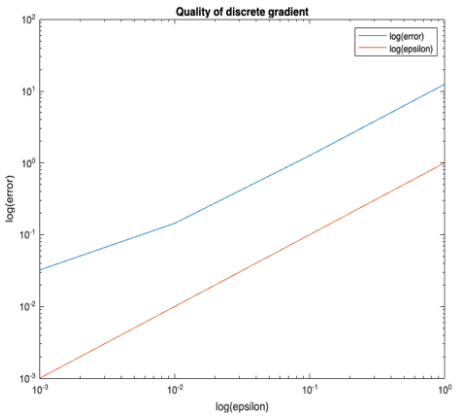
To check the quality of the discrete gradient, we will use the following formula:

$$\left| \frac{\hat{J}(u + \varepsilon d_u) - \hat{J}(u)}{\varepsilon} - \nabla \hat{J}(u)^T d_u \right| = O(\varepsilon)$$

For the different perturbations, the term at the left side of the equation should be parallel to the perturbations. We have plotted the terms as you can see below which confirms that the quality of our discrete gradient method is acceptable.

```
lambda_N = 0;
bheta = 1/2;
epsilons = [1, 0.1, 0.01, 0.001];
values = zeros(1,length(epsilons));
for i = 1:length(epsilons)
    Jh_norm = cost(u0, f, t0, tF, x0, h, rho,c,p,q,upper_bound_for_population);
    Jh_eps = cost(u0+epsilons(i).*du0, f, t0, tF, x0, h, rho,c,p,q,upper_bound_for_population);
    gr_Jh = gr_cost(u0, f, t0, tF, x0, h, rho,c,p,q, lambda_N, derr_g(), bheta,upper_bound_for_population);
    t_du0 = transpose(du0);
    value = ((Jh_eps-Jh_norm)/epsilons(i)) - (gr_Jh*t_du0);
    values(1,i) = value;
end

loglog(epsilons,values); hold on; loglog(epsilons,epsilons); hold off;
title('Quality of discrete gradient');
xlabel('log(epsilon)');
ylabel('log(error)');
legend('log(error)','log(epsilon)');
```



4.Optimization Solver

The projected gradient method is a widely-used optimization algorithm designed to address constrained optimization problems. It extends the steepest descent method by incorporating an additional step to enforce problem constraints.

The algorithm begins by initializing an initial solution guess. At each iteration, it calculates the steepest descent direction by computing the negative gradient of the objective function. This direction indicates the most significant reduction in the objective function value.

However, the projected gradient method goes beyond the steepest descent method by employing a projection step. This step ensures that the obtained solution remains within the feasible set defined by the problem's constraints. By utilizing a projection operator, the algorithm projects the current solution onto the feasible set, effectively implementing the constraints.

The step size is determined by applying the generalized Armijo's condition.

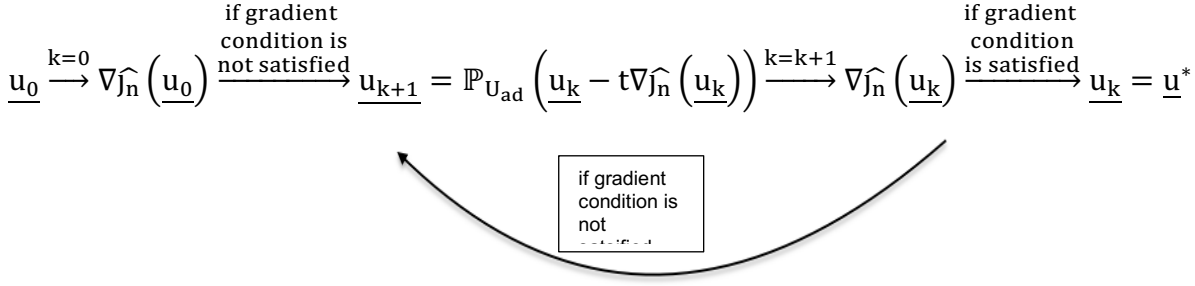
$$\boxed{\underline{u_k}(t) := \mathbb{P}_{U_{ad}}(\underline{u_k} - t\nabla J(\underline{u_k}))}$$

$$\text{Generalized Armijo's Condition: } J\left(\underline{u_k}(t)\right) \leq J\left(\underline{u_k}\right) - \frac{\alpha}{t}\left\|\underline{u_k} - \underline{u_k}(t)\right\|_2^2$$

This condition verifies that the objective function value decreases sufficiently along the descent direction, taking into account a user-defined step size factor. If the condition is not met, the step size is decreased until the condition holds, balancing the trade-off between convergence speed and accuracy.

The method continues iterating until a convergence criterion is satisfied, such as reaching a specified tolerance or a maximum number of iterations. Ultimately, the resulting solution represents an approximate optimal solution that satisfies the imposed constraints.

In summary, the projected gradient method, enhanced by the generalized Armijo's condition, is a robust approach for solving constrained optimization problems. By combining the benefits of the steepest descent method with constraint enforcement through projection, it efficiently and accurately determines optimal solutions within the feasible set. Outlined flow given below.



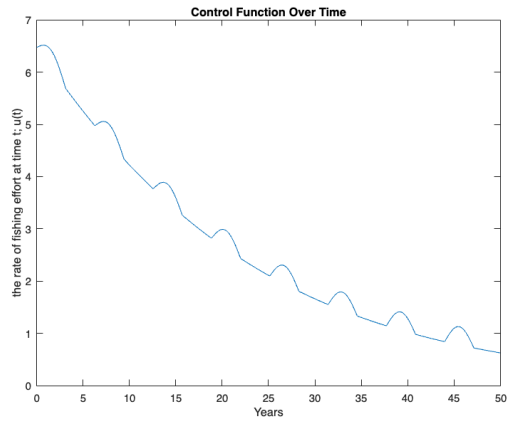
Used MATLAB functions given in the appendix.

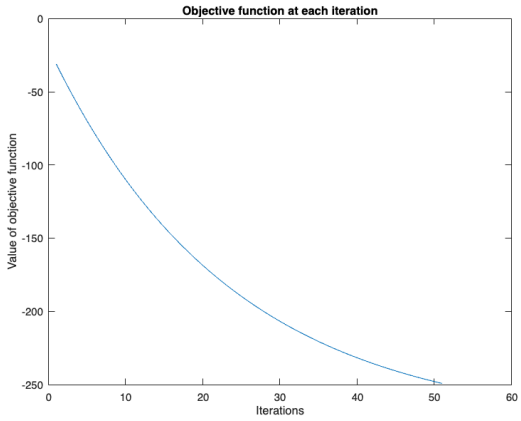
5.Numerical experiments

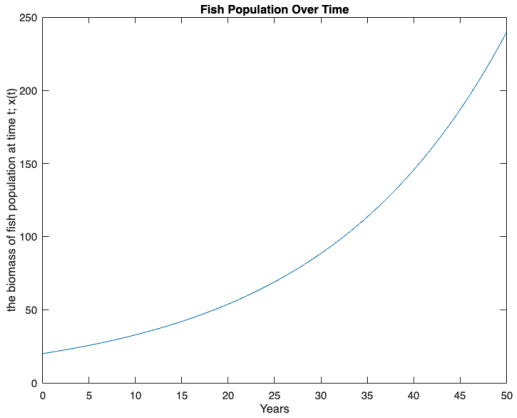
Case 1 (Base Scenario)

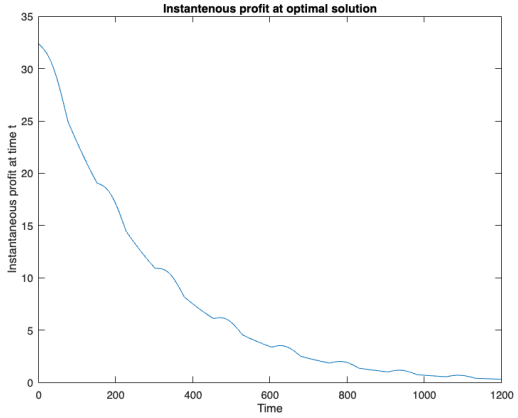
In the first case, we examined the output of the case whose parameters described in the introduction part of the report.

	Parameter	Value
1	'starting time (year)'	0
2	'ending time (year)'	50
3	'timestep'	1/24
4	'inital biomass of fish population (tons) '	20
5	'the catchability coefficient (1/tons)'	1/10000
6	'the discount rate'	1/10
7	'the unit price of landed fish (money/tons)'	3000
8	'the unit cost of effort (money/tons)'	1
9	'upper bound for population'	900
10	'the natural growth function'	x/20
11	'alpha'	1/2
12	'bheta'	1/2
13	'upperU'	900
14	'tol'	1/1000000









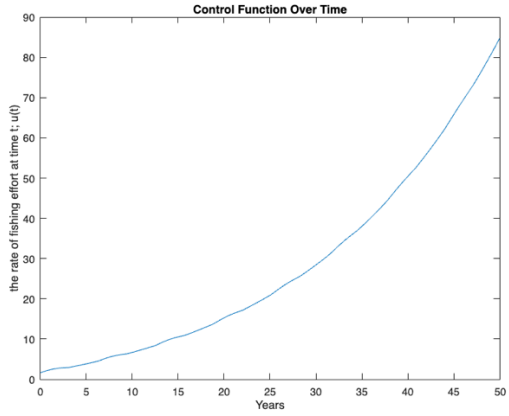
Comment:

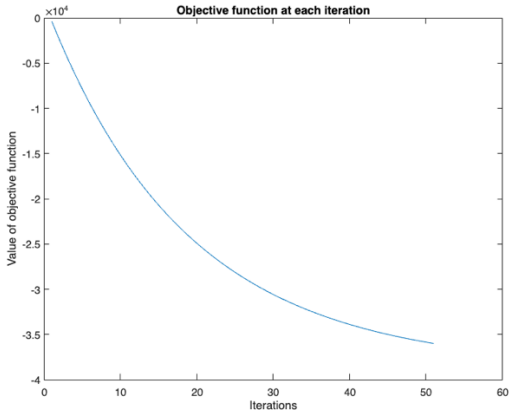
In this case, we simulate the dynamics of a fish population over a time span of 50 years. The initial fishing effort is determined by the sine function, resulting in periodic variations. The model incorporates various parameters given above and the output consists of optimal control function, biomass of the fish population over time in the optimal scenario, changes in the value of cost function at each iteration of projected gradient search, and instantaneous profit (present value) at each year in the optimal scenario.

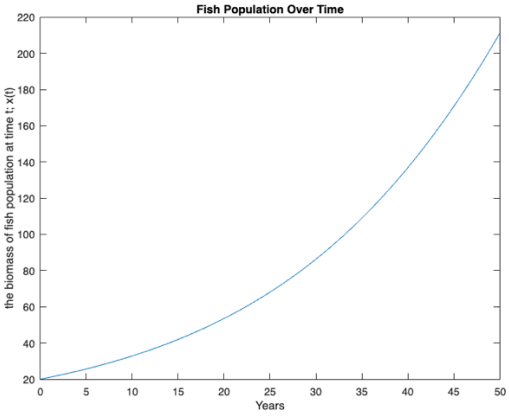
Case 2 (0 discount rate)

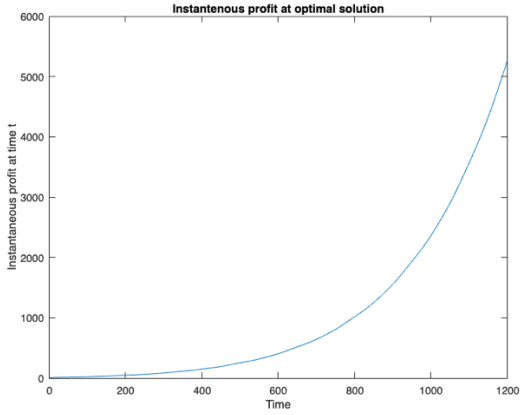
In the second case, we analyzed the output of a scenario with a discount rate (rho) set to 0. In this case, we anticipated observing a not descending trend in the instantaneous profit, as the value of money earned in later years has the same monetary value as the money earned in earlier years.

	Parameter	Value
1	'starting time (year)'	0
2	'ending time (year)'	50
3	'timestep'	1/24
4	'inital biomass of fish population (tons) '	20
5	'the catchability coefficient (1/tons)'	1/10000
6	'the discount rate'	0
7	'the unit price of landed fish (money/tons)'	3000
8	'the unit cost of effort (money/tons)'	1
9	'upper bound for population'	900
10	'the natural growth function'	x/20
11	'alpha'	1/2
12	'bheta'	1/2
13	'upperU'	900
14	'tol'	1/1000000









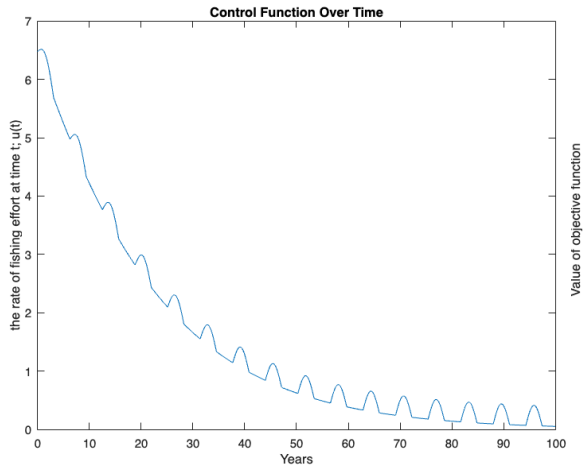
Comment:

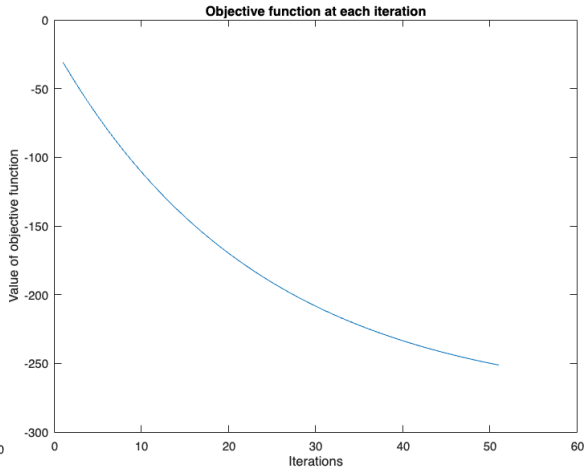
As expected, the instantaneous profit doesn't decrease over time since the money doesn't lose the its value. Also, optimal value of the cost function is improved compared to base scenario. In addition, the control function doesn't have any fluctuations, unlike the first scenario, and increase monotonically.

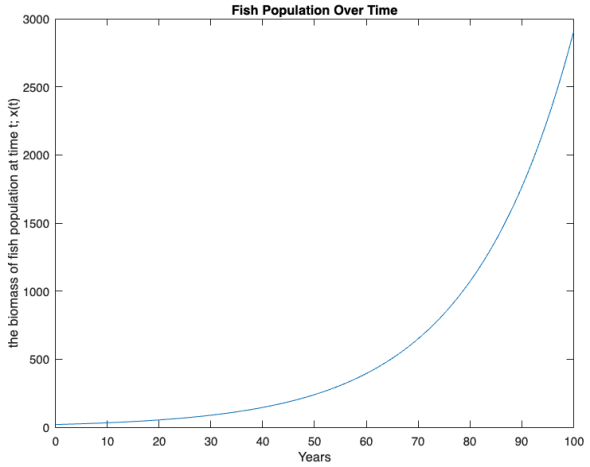
Case 3 (longer time period)

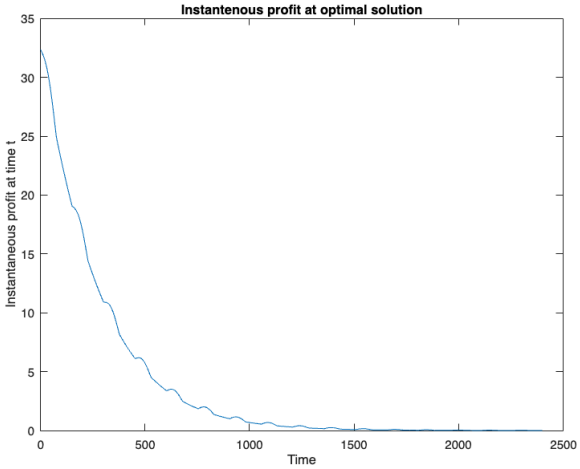
In the third case, we analyzed the output of a scenario with the tF = 100 instead of 50. We expected to obtain the similar patterns in the first case just with the different values.

	Parameter	Value
1	'starting time (year)'	0
2	'ending time (year)'	100
3	'timestep'	1/24
4	'inital biomass of fish population (tons) '	20
5	'the catchability coefficient (1/tons)'	1/10000
6	'the discount rate'	1/10
7	'the unit price of landed fish (money/tons)'	3000
8	'the unit cost of effort (money/tons)'	1
9	'upper bound for population'	900
10	'the natural growth function'	x/20
11	'alpha'	1/2
12	'bheta'	1/2
13	'upperU'	900
14	'tol'	1/1000000









Comment:

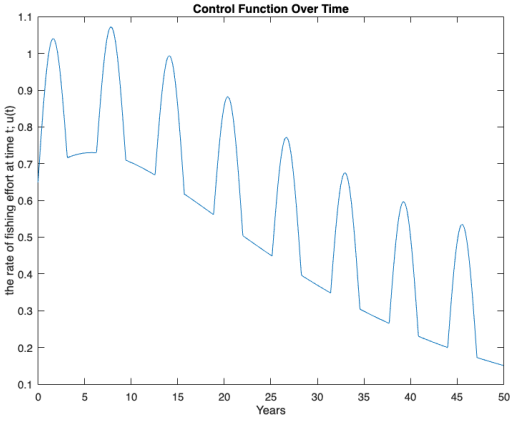
As expected, structures of the plots are similar to the first case but the end values are different at the end of the whole life period.

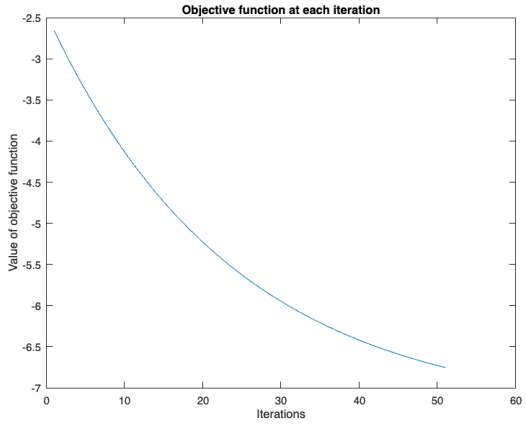
Case 4 (different initial biomass of population)

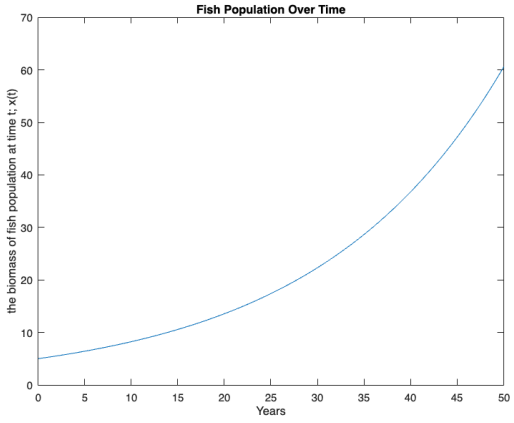
In the fourth case, we will examine the patterns caused by the changes in the initial biomass of the population. We first demonstrated the decreased initial biomass (x0 = 5); and then, increased initial biomass (x0 = 100) as two separate scenarios.

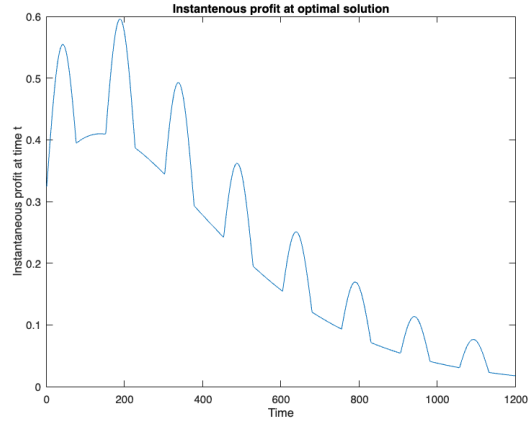
4.1 (x0 = 5)

	Parameter	Value
1	'starting time (year)'	0
2	'ending time (year)'	50
3	'timestep'	1/24
4	'inital biomass of fish population (tons) '	5
5	'the catchability coefficient (1/tons)'	1/10000
6	'the discount rate'	1/10
7	'the unit price of landed fish (money/tons)'	3000
8	'the unit cost of effort (money/tons)'	1
9	'upper bound for population'	900
10	'the natural growth function'	x/20
11	'alpha'	1/2
12	'bheta'	1/2
13	'upperU'	900
14	'tol'	1/1000000







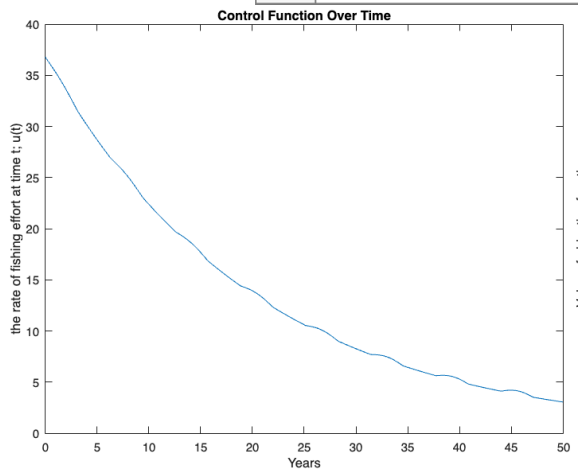


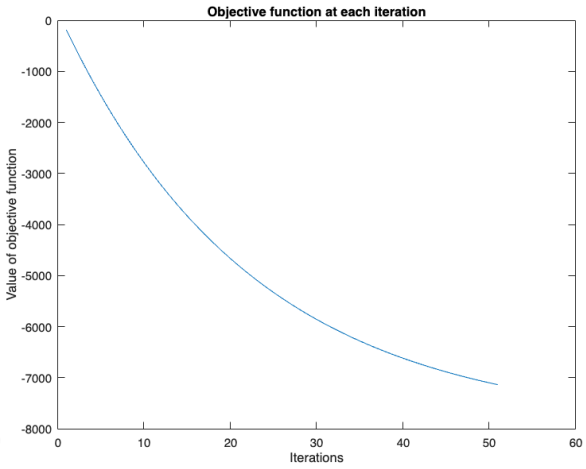
Comment:

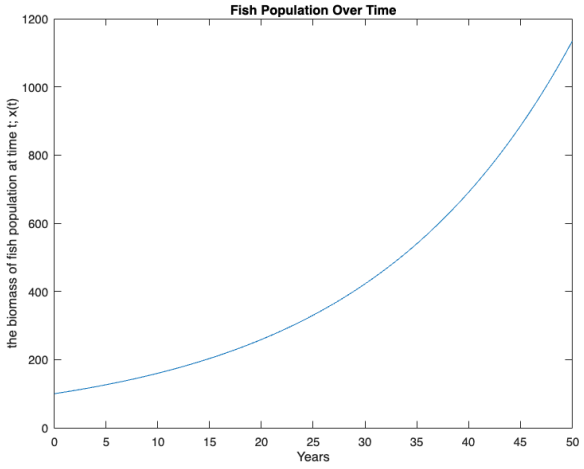
Due to the reduced initial population value, the control function exhibits lower values while maintaining a plot pattern similar to the first case. Furthermore, the functions exhibit more pronounced fluctuations. This can be attributed to the smaller values, as even slight changes in these values appear relatively larger in scale.

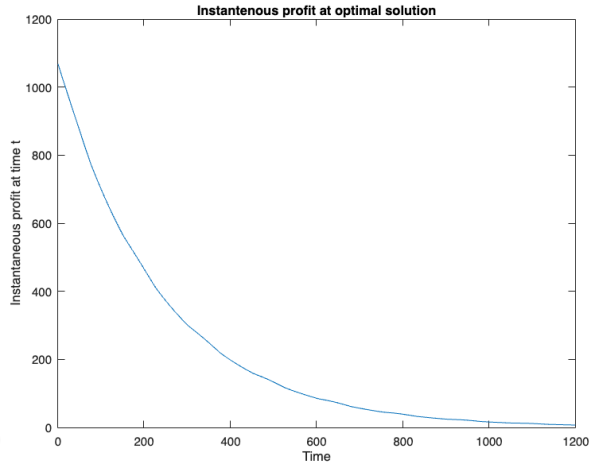
4.2 (x0 = 100)

	Parameter	Value
1	'starting time (year)'	0
2	'ending time (year)'	50
3	'timestep'	1/24
4	'inital biomass of fish population (tons) '	100
5	'the catchability coefficient (1/tons)'	1/10000
6	'the discount rate'	1/10
7	'the unit price of landed fish (money/tons)'	3000
8	'the unit cost of effort (money/tons)'	1
9	'upper bound for population'	900
10	'the natural growth function'	x/20
11	'alpha'	1/2
12	'bheta'	1/2
13	'upperU'	900
14	'tol'	1/1000000









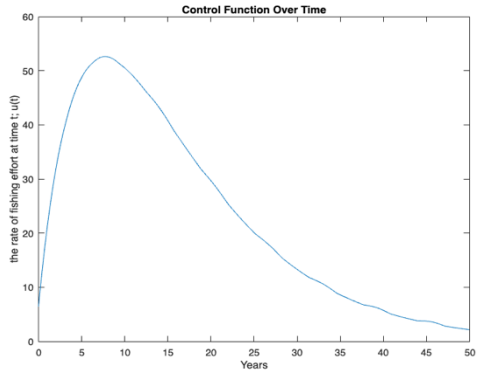
Comment:

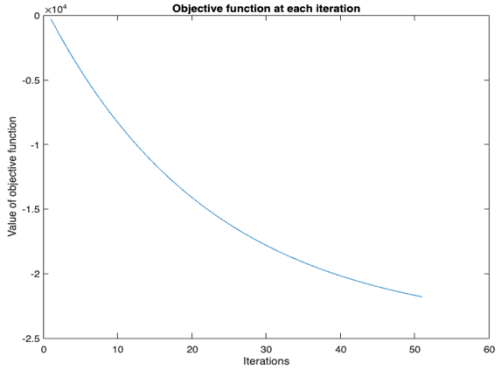
Due to the increased initial population value, the control function exhibits lower values while maintaining a plot pattern similar to the first case. It appears that the fluctuations are relatively smaller in scale, giving the impression that they have decreased. This effect can be attributed to the increased magnitude of the table.

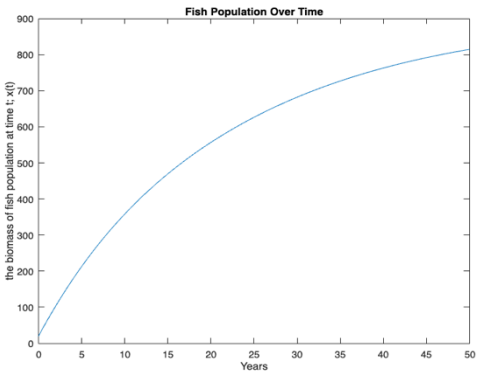
Case 5 (upper bound for population)

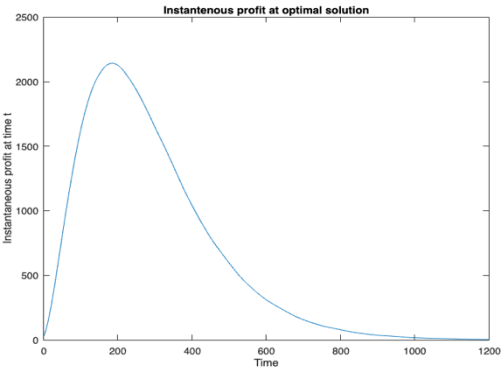
In this case, we will enforce an upper bound for biomass of fish population. The upper bound is 900 (tons) as described in the introduction part of the report. We implement the upper bound by using the decreased growth function g(x) = (upper_bound_of_population – x) * 0.05. As a result of the change in the growth function, population will decrease (die) if it exceeds the upper bound. Furthermore, in this case, it is worth noting that the growth rate tends to be higher as the population decreases, which presents a somewhat controversial aspect.

	Parameter	Value
1	'starting time (year)'	0
2	'ending time (year)'	50
3	'timestep'	1/24
4	'inital biomass of fish population (tons) '	20
5	'the catchability coefficient (1/tons)'	1/10000
6	'the discount rate'	1/10
7	'the unit price of landed fish (money/tons)'	3000
8	'the unit cost of effort (money/tons)'	1
9	'upper bound for population'	900
10	'the natural growth function'	45 - x/20
11	'alpha'	1/2
12	'bheta'	1/2
13	'upperU'	900
14	'tol'	1/1000000









Comment:

As depicted in the visualization, the population remains below the upper bound of 900 tons. Additionally, compared to the case of linear increasing growth, the objective function value has improved. This improvement can be attributed to the faster increase in population at the beginning of the simulation, as the initial population is relatively low (x0 = 20 tons). The combination of faster growth and the discount rate leads to increased initial fish sales, resulting in higher profitability.

6.Conclusion & Remarks

In conclusion, this simulation and report have provided an analysis of the sole owner fishery model. The model offers insights into the dynamics of fish population, fishing effort, and profitability in a sustainable fishing environment. Through the discretization of the Ordinary Differential Equation, we obtained a numerical approximation that enabled the study of the system's behavior over time. The optimization solver, in conjunction with the optimality systems, allowed us to optimize the fishing effort and maximize the objective function while considering various constraints. The numerical experiments conducted in different scenarios (Case 1 to Case 5) provided valuable insights into the sensitivity of the model to changes in parameters such as discount rate, time period, initial biomass, and population upper bound. These experiments demonstrated the model's versatility and its ability to capture and analyze various real-world scenarios.

Remarks:

- Carrying capacity for fish population is ignored except for the last case.
- Constant cost and price over time but monetary value can change due to discount rate.
- There are several assumptions regarding the parameters and functions such as growth function, upper bound for fishing effort and carrying capacity, initial population, discount rate, price, and cost, which have been explained in the introduction. To fully understand the simulation, it is important not to overlook these assumptions. Different assumptions may lead to different results.

7.References

Sethi, S. P. (2018). Optimal Control Theory: Applications to Management Science and Economics. Almanya: Springer International Publishing.

https://ir.library.oregonstate.edu/downloads/2z10wr225

http://www.historyoffishing.com/fishing-facts/types-of-fishing-techniques/

https://www.worldatlas.com/articles/the-world-s-most-common-types-of-fish.html

https://www.selinawamucii.com/insights/prices/united-states-of-america/carp-fish/

https://www.researchgate.net/figure/Average-observed-fish-density-kg-m-3-by-depth-in-sea-caged-Atlantic-salmon_fig3_233773290

8.Appendix

Explicit Euler

```
function [th, uh] = explicit_euler(u, f, t0, tF, y0,  h)
    th = t0:h:tF;
    N = length(th);
    uh = zeros(1,N);
    uh(1) = y0;
    for n = 1:N-1
        uh(n+1) = uh(n) + h*f(th(n),uh(n),u(n));
    end
end
```

Jhat

```
function j_hat = Jhat(u,th,uh,h,rho,c,p,q,bheta,upper_bound_for_population)
    N = length(th); % Nh+1
    s1 = 0;
    s2 = 0;
    for i = 0:N-2
        s1 = s1+ exp(-rho*th(i+1)) * (c - p * q *uh(i+1)) * u(i+1);
    end
    for i = 0:N-2
        s2 = s2 + (u(i+1))^2;
    end
    j_hat =  (h * s1) + (bheta/2 * h * s2);
end
```

Cost (Compact function consists of EE and Jhat)

```
function Jh = cost(u, f, t0, tF, y0, h, rho,c,p,q,bheta,upper_bound_for_population)
    [th, uh] = explicit_euler(u, f, t0, tF, y0,  h);
    Jh = Jhat(u,th,uh,h,rho,c,p,q,bheta,upper_bound_for_population);
end
```

Adjoint Equation

```
function lambdas = lambda_array(u,th,uh,h,rho,p,q, lambda_N,
derr_g,upper_bound_for_population)
    N = length(th);
    lambdas  = zeros(1,N);
    lambdas(N) = lambda_N;
    for i = N-1:-1:1
        lambdas(i) = lambdas(i+1) * (1 + h * (derr_g - q * u(i))) + h * (p * q *
u(i) * exp(-rho * th(i) ) );
    end
end
```

Gradient Jhat

```
function gr_j_hat = gradient_j_hat(u, th, uh, h, rho, c, p, q, lambdas, bheta)
    N = length(th);
    gr_j_hat = zeros(1,N - 1);
    for i = 1:N-1
        gr_j_hat(i) = h * (bheta * u(i) + exp(-rho * th(i)) * (c - p * q *uh(i)) +
q * uh(i) * lambdas(i+1));
    end
end
```

Gradient Cost (Compact function consists of EE, Adjoint Equation, and Gradient Jhat)

```
function [gr_Jh] = gr_cost(u, f, t0, tF, y0, h, rho,c,p,q, lambda_N, derr_g,
bheta,upper_bound_for_population)
    [th, uh] = explicit_euler(u, f, t0, tF, y0,  h);
    lambdas = lambda_array(u,th,uh,h,rho,p,q, lambda_N,
derr_g,upper_bound_for_population);
    gr_Jh = gradient_j_hat(u, th, uh, h, rho, c, p, q, lambdas, bheta);
end
```

Projection

```
function new_u = projecting_u(u, lower_bound, upper_bound)
        u = max(u,lower_bound);
        u = min(u,upper_bound);
        new_u = u;
end
```

Linesearch (backtracking)

```
function t = armijo_back(data,u,d,alpha,itmax,upperU)
    % armijo_back: function that performs a line-search backtracking strategy based
on the Armijo rule
    % Input :
    % data  : structure that contains all the data of the problem
    % u, d  : current iteration and direction
    % alpha : parameter for the Armijo condition
    % itmax : maximum number of iterations
    t=1;
    J = data.J;
    f = @(a) J(projecting_u(u+a.*d, 0,upperU));
    f0 = f(0);
    it = 1;
    while f(t) > f0 - (alpha/t) * (norm(u - projecting_u(u+t.*d, 0,upperU)))^2 &&
it<itmax % check condition
        t = (1/2)^it; % backtrack
        it = it+1; % update it end
    end
end
```

Projected Gradient

```
function [u,opt,it,um,optm,cst] = descentmethod(data,tol,itmax,u0,upperU)
    % descentmethod: function that performs a (projected) gradient method
    % Input :
    % data : structure that contains all the data of the problem % tol : tolerance
for termination
    % itmax : maximum number of iterations
    % u0 : starting point
    u0 = projecting_u(u0, 0, upperU);
    u = u0;
    J = data.J;
    dJ = data.dJ;
    linesearch = data.linesearch;
    dJu = dJ(u0);
    opt = 1; it = 0;
    um = u0; optm=1;
    cst = J(u);
    while opt>tol && it<itmax
        d = -dJu;
        t = linesearch(data,u,d);
        u = u+t*d;
        u = projecting_u(u, 0, upperU);
        dJu = dJ(u);
        next_u = u-1*dJu;
        next_u = projecting_u(next_u, 0, upperU);
        opt = norm(u - next_u);
        if nargout > 3
            um = [um,u]; optm = [optm,opt]; cst = [cst, J(u)];
        end
        it = it+1;
        fprintf('it= %3d J(u)= %10.6e |dJ(u)|= %10.6e\n', it, J(u), opt);
    end
end
```

Profit

```
function profits = profit(p, q, c, rho, th, uh, u)
    N = length(th);
    profits = zeros(1,N-1);
    for i = 1:N-1
        profits(i) = exp(-rho*th(i)) * (p * q *uh(i)-c) * u(i);
    end
end
```

Fishery Model (Main simulation script)

```
%Starting/ending (year)
t0 = 0;
tF = 50;

%timestep
h = 1/24;
time = t0:h:tF;

%Initial value of biomass at time t0 (tons)
x0 = 20;

upper_bound_for_population = 900;

%the natural growth function
syms x real
g =@(x) 0.05*(x);
sym_derr = diff(g(x));
derr_g = matlabFunction(sym_derr);


%the rate of fishing effort at time t(tons/time)
u0 = sin(time(1:end-1)); %ones(1,1 + (tF-t0)/h).*5;
%u0 = zeros(1,length(time(1:end-1)));
du0 = cos(time(1:end-1));

%the catchability coefficient(1/tons)
q = 0.0001;

%the discount rate(no unit)
rho = 0.1;

%the unit price of landed fish(money/tons)
p = 3000;

%the unit cost of effort(money/tons)
c = 1;

%derivative of x w.r.t. t
f =@(t,x,u) g(x) - q.*u.*x;

lambda_N = 0;
alpha = 1/2;
bheta = 1/2;

upperU = 900;
data.J =@(u) cost(u, f, t0, tF, x0, h,
rho,c,p,q,bheta,upper_bound_for_population);
data.dJ = @(u) gr_cost(u, f, t0, tF, x0, h, rho,c,p,q, lambda_N, derr_g(),
bheta,upper_bound_for_population);

data.linesearch = @(data,u,d) armijo_back(data,u,d,alpha,50,upperU);
tol = 1e-6;
[u,opt,it,um,optm,cst] = descentmethod(data,tol,50,u0,upperU);
```

```
s = time(1:end-1);
plot(s,u)
title('Control Function Over Time');
xlabel('Years');
ylabel('the rate of fishing effort at time t; u(t)');

[th, uh] = explicit_euler(u, f, t0, tF, x0,  h);
plot(th,uh);
title('Fish Population Over Time');
xlabel('Years');
ylabel('the biomass of fish population at time t; x(t)');

plot(cst);
title('Objective function at each iteration');
xlabel('Iterations');
ylabel('Value of objective function');

profits = profit(p, q, c, rho, th, uh, u);
plot(profits)
title('Instantenous profit at optimal solution');
xlabel('Time');
ylabel('Instantaneous profit at time t');

% Define parameter names and values
parameterNames = {'starting time (year)', 'ending time (year)', 'timestep',
'inital biomass of fish population (tons) ', 'the catchability coefficient
(1/tons)', 'the discount rate', 'the unit price of landed fish (money/tons)',
'the unit cost of effort (money/tons)', 'upper bound for population', 'the
natural growth function', 'alpha', 'bheta', 'upperU', 'tol'};
parameterValues = [t0, tF, h, x0, q, rho, p, c, upper_bound_for_population,
g(x), alpha, bheta, upperU, tol];

% Create table
paramTable = table(parameterNames', parameterValues', 'VariableNames',
{'Parameter', 'Value'})
```