Numerical Approach to King and Roughgarden 1982

To test our approach, I tried to write code to the optimal control problem from King and Roughgarden 1982 using the numerical methods we came up with in the spring. King and Roughgarden used the Poyntragin maximum principle to obtain an analytic solution to their optimal control problem. If I've read Clark (19XX) correctly, the maximum principle is useful for low-dimensional problems but numerical methods are often used for higher dimensional problems. However, both approaches involve the following components:

- a cost or loss function (J) to be optimized (minimized or maximized)
- a system of differential equations $\dot{\boldsymbol{x}}$
- constraints u

King and Roughgarden propose that the long-term, optimal reproductive strategy will maximize the geometric mean of reproductive success. They thus propose that the function that should be optimized is the expectation of the log of fitness, $J = (1/T)\int_0^T \log(x_2)$. They had a system of equations

$$\begin{aligned} \dot{x_1} &= u(t)x_1 \\ \dot{x_2} &= (1-u(t))x_1 \end{aligned} \tag{1}$$

subject to

$$\begin{aligned} 0 \leq & u(t) \leq 1 \\ 0 < & x_1 \\ 0 \leq & x_2 \end{aligned} \tag{2}$$

I implemented the constraints on the rate u as $u \leq 1$ as $u - 1 \leq 0$ and that u is positive as:

$$A_{11} = \begin{bmatrix} -1 & 0 & \dots & 0 \\ 0 & -1 & \ddots & \vdots \\ \vdots & \ddots & -1 & 0 \\ 0 & \dots & 0 & -1 \end{bmatrix}_{t \times t} \tag{3}$$

$$A_1 = \begin{bmatrix} A_{11} \end{bmatrix}_{t \times t} \tag{4}$$

$$b_1 = \begin{bmatrix} -1 & -1 & \dots & -1 \end{bmatrix}_{1 \times t} \tag{5}$$

At each time step, this constraint specifies $-1 \times u - 1 \geq 0$, which is equivalent to $u \leq 1$. We implemented the constraint that u be greater than or equal to 0 as:

$$A_2 = \begin{bmatrix} 1 & 0 & \dots & 0 \\ 0 & 1 & \ddots & \vdots \\ \vdots & \ddots & 1 & 0 \\ 0 & \dots & 0 & 1 \end{bmatrix}_{t \times t} \tag{6}$$

$$b_2 = \begin{bmatrix} 0 & 0 & \dots & 0 \end{bmatrix}_{1 \times t} \tag{7}$$

At each time step, this constraint specifies $1 \times u - 0 \geq 0$, which is equivalent to $u \geq 0$. Together

$$\boldsymbol{A} = \begin{bmatrix} A_1 & A_2 \end{bmatrix}_{2t \times t} \tag{8}$$

and

$$\boldsymbol{b} = \begin{bmatrix} b_1 & b_2 \end{bmatrix}_{1 \times 2t} \tag{9}$$

These matrices define the following linear inequality constraints:

$$A\boldsymbol{\theta} - \boldsymbol{b} \geq 0 \tag{10}$$

I use these constraints to optimize the cost function in King and Roughgarden 1982 [$J = \int_0^T log(x_2)$]. We do so by defining linear inequality constraints, writing a function for the ODEs, and nesting the function for the ODEs within a function that performs optimization.

The system of differential equations $\dot{\boldsymbol{x}}$ for $0 \leq t \leq T$ is written as a function that computes the values of derivatives in the ODE system at time t:

```
## Function that computes values of derivatives in the ODE system
## Takes the time step, system of differential equations, parameters
derivs = numeric(2);
control <- function(times0,y,parms,f1,...) {

  # x1 and x2 are the two entries in y (ode)
  x1=y[1];

  # values calculated by the interpolated function at different time points
  u <- f1(times0);

  derivs = c(u*x1,(1-u)*x1)
  return(list(derivs));
}
## Compiles the function control()
control=cmpfun(control);
```

The linear inequality constraints $A\boldsymbol{\theta} - \boldsymbol{b} \geq 0$ are written as:

```
## Number of time steps t
## Also used in control() and optim()
t = 10;

## Block 1
## Constraint that u-1 <= 0
A1 = matrix(0,nrow=t,ncol=t);
diag(A1) = -1; c1 = rep(-1,t);

## Block 2
## Constraint u>=0
A2 = matrix(0,nrow=t,ncol=t);
diag(A2) = 1; c2 = rep(0,t);

## Create matrix A and vector b
Amat = rbind(A1,A2);
bvec = c(c1,c2);
```

The ODE and optimization function both require initial conditions. Here, we set starting values for the ODE ($\boldsymbol{x}(0)$), define the time steps at which the ODE is solved, and an initial guess at the optimal control $u(t)$. In general, we would also specify any parameters in the model at this point but this is an empty vector because the ODE does not take additional parameters.

```
## Starting values for ODE
yA=c(x1=1,x2=45);
## Time steps for ODE
seq_length = 100;
timesA=seq(0,t-1,length=seq_length);
## Parameters in the ODE
parmsA=c();

## Initial guess at the control function
v_in = rep(.1,t);
```

Finally, before we begin optimization we check to make sure that our initial guess satisfies the specified constraints.

```
## Check to make sure initial guess is acceptable
## If FALSE ok to proceed
any(Amat %*% v_in - bvec <= 0)
```

Broadly, we implement optimization by using R functions, `optim` and `constrOptim`, that take a function and parameters over which to optimize that function. In our case, the parameters

to optimize are the values of the control trajectory $u(t)$. The function we want to optimize is the loss function J subject to the ODE and constraints.

At each time step in the optimization procedure these functions takes a vector $u(t)$ (either an initial guess or output from a previous iteration of optimization) and uses `approxfun` to create an interpolated function that approximates the trajectory of $u(t)$.

```
u_fun <- approxfun(0:(t-1),par[1:t],rule=2);
```

This interpolated function for $u(t)$ is then used by `ode(...)` at each time step to solve the ODE with the initial values specified above.

```
    out = ode(y=y0,
           times=times0,
           func=control,
           parms=parms0,
           atol=1e-7,
           f1=u_fun);
```

The solution to the ODE is a matrix with values for x_1 and x_2 at each time step. We use `splinefun(...)` to obtain a function that interpolates the values of x_2 at each time step. We first then augment these values beyond the range of $(0, T)$ so that the function behaves smoothly towards the ends of the range.

```
    R_vec = out[,3];
    R_vec_plus = c(0,R_vec,max(R_vec));
    timesA_plus = c(0-t,times0,2*t);
    f_fun <- splinefun(timesA_plus,R_vec_plus);
```

We now have a function `f_fun(...)` that calculates the value x_2 at each time step. We use this to minimize $J = 1/T \int_0^T log(x_2)$.

This is a two step process. If the interpolated function (`f_fun()`) returns values outside of the range of x_2 in $(0, T)$, the code gives a large positive value so that the control trajectory giving this particular interpolated function is not considered optimal. Otherwise, the code returns the negative of the integral of the log of reproductive biomass (x_2) from 0 to T (`integrate(...)`) divided by the season length T. This value is what's being optimized by R's optimization routines.

```
## Divide (0,T) into a grid
mesh <- seq(0.0001, t ,length.out=1000);

## Function for the integrand
integrand <- function(x){
  log(f_fun(x))
```

```
}

## Reject control trajectory if x2 <= 0
## Return expected value of integral on (0,T)
min = if(any(f_fun(mesh)<=0)) 1000 else
    -1*((integrate(integrand,lower=0.00001,upper=t)$value)/(t))
```

In our code, we first use `optim` to improve on our initial guess at the control trajectory $u(t)$. The function `penalized_fun` applies the optimization procedure and additionally adjusts the control trajectory $u(t)$ to lie within $(0, T)$ before interpolation.

```
## initial optimization
fit <- optim(par=v_in, fn=penalized_fun, method="BFGS",
            control=list(maxit=2500,trace=1,REPORT=5))

theta=fit$par;

## adjust the us to be greater than or equal to 0
for(j in 0:1000) {
    theta=fit$par + (j/1000)*(v_in - fit$par)
    z = min(Amat %*% theta - bvec)
    if(z>=0) break
}
```

Once we've improved our initial guess, we use `constrOptim` to carry out the optimization procedure subject to the linear inequality constraints specified by $A\boldsymbol{\theta} - \boldsymbol{b} \geq 0$. This function optimizes a control vector $u(t)$: `theta` subject to the constraints imposed by A: `ui=Amat` and $\boldsymbol{b}$: `ci=bvec`. Additionally, we provide the function with a function giving the gradient of f calculated by centered difference [ref?].

```
fit<-constrOptim(theta = theta,
            f = optim_fun,
            grad = optim_grad,
            ui = Amat,
            ci = bvec,
            method="BFGS",
            control=list(maxit=2500,trace=1,REPORT=1),
            outer.iterations = 20,
            outer.eps = 1e-06)
```

We apply these functions in sequence to solve our optimization problem. We can then plot the optimal trajectory of $u(t)$ subject to the ODE, constraints, and initial conditions. We can also plot the solution to the ODE under the optimal trajectory of $u(t)$.