Laboration: Analytical Manipulations in Bio Informatics

Simon Sigurdhsson

October 20, 2010

1 A simple biological system

Consider a simple biological system composed of two different biological subsystems; one component with three parallel coupled components, all Weibull(1,$\frac{1}{2}$)-distributed, and one single fourth exp($\frac{1}{2}$)-distributed component serially coupled to the first three (see Figure 1). It is of interest to compute expected lifelength of this system, among other things.

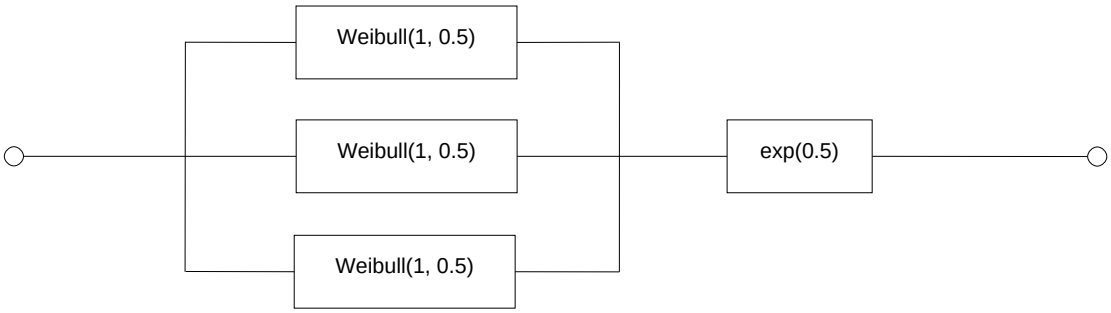

Figure 1: The simple biological system examined in this section

1.1 Expected biological life length

For the first part of the system, we have a biological survival function $(1-(1-R_W(t))^3)$, where $R_W(t) = 1 - F_W(t)$ and $F_W(t)$ is the Weibull(1,$\frac{1}{2}$) distribution function. The second part contributes a factor $R_e(t) = 1 - F_e(t)$, where $F_e(t)$ is the exp($\frac{1}{2}$) distribution function. Multiplying these gives us the biological survival function for the whole system:

$$R(t) = R_e(t)\left(1 - (1 - R_W(t))^3\right) = (1 - F_e(t))\left(1 - F_W(t)^3\right)$$

We can then use this biological survival function to calculate the expected biological life length easily, using the identity given in Theorem 6.3 of the assignment:

$$\mathbf{E}\{T\} = \int_0^\infty R(t)\,\mathrm{d}t = \int_0^\infty (1 - F_e(t))\left(1 - F_W(t)^3\right)\,\mathrm{d}t = \int_o^\infty e^{-\frac{t}{2}}\left(1 - \left(1 - e^{-\sqrt{t}}\right)^3\right)\,\mathrm{d}t$$

Clearly, this is no integral we want to calculate analytically, at least not by hand. While it *can* be calculated analytically, the indefinite integral contains three instances

of the error function and is very messy. Even the definite (improper) integral looks horrible in its analytic form:

$$\int_0^\infty R(t)\,\mathrm{d}t = -3\sqrt{2e\pi}\mathrm{erfc}\left(\frac{1}{\sqrt{2}}\right) - 3e^2\sqrt{2\pi}\left(e^{5/2}\mathrm{erfc}\left(\frac{3}{\sqrt{2}}\right) - 2\mathrm{erfc}\left(\sqrt{2}\right)\right) + 2$$

It should be noted that this integration was performed in Mathematica, along with a proper evalution of the error function to recieve a numerical value of the expected biological life length:

```
In[1]:= R[t_] := Exp[-t/2](1-(1-Exp[-Sqrt[t]])^3)
In[2]:= Integrate[R[t], {t, 0, Infinity}]
Out[2]= (* See equation *)
In[3]:= N[%]
Out[3]= 1.29481
```

Thus, the expected biological life length of this small system is 1.2948 time units.

It may also be of interest to know wether this system is BIBFR, BDBFR, or neither. To find this out, we simply plot the biological death rate, given by $r(t) = \frac{\mathrm{d}}{\mathrm{d}t}\ln(R(t))$. This function, like R, *can* be expressed analytically, but it's just as messy — we'll be happy to simply plot it:

```
In[4]:= r[t_] := -D[Log[R[t]]]
In[5]:= Plot[r[t], {t, 0, 10}]
Out[5]= -Graphics-
In[6]:= Export["fig1.eps", %]
Out[6]= fig1.eps
```

As seen in Figure 2, the biological death rate is steadily increasing (and almost linearly so). This is not surprising, given the exponential nature of $R(t)$ and the logarithm included in $r(t)$. We can conclude that this biological system is BIBFR.

1.2 Component death

Continuing to examine the same biological system, we may also want to know what componens are most likely to break. Specifically, we may be interested in the probability that the last component (the $\exp(\frac{1}{2})$-distributed one) is the component that causes the system to fail. This probability is equivalent to the probability that at least one of the other three componens survive longer than the one we're interested in, since they're parallel coupled:

$$\mathbf{P}\{\text{Component 4 causes biological death}\} = \mathbf{P}\{\max(T_1, T_2, T_3) > T_4\},$$
$$\text{where } T_i \text{ is the biological lifelength of component } i.$$

We can also rewrite this probability to something slightly more useful, making use of basic probability distribution theory (transforming it into an integral containing the probability distribution function of the fourth component, as opposed to the cumulative

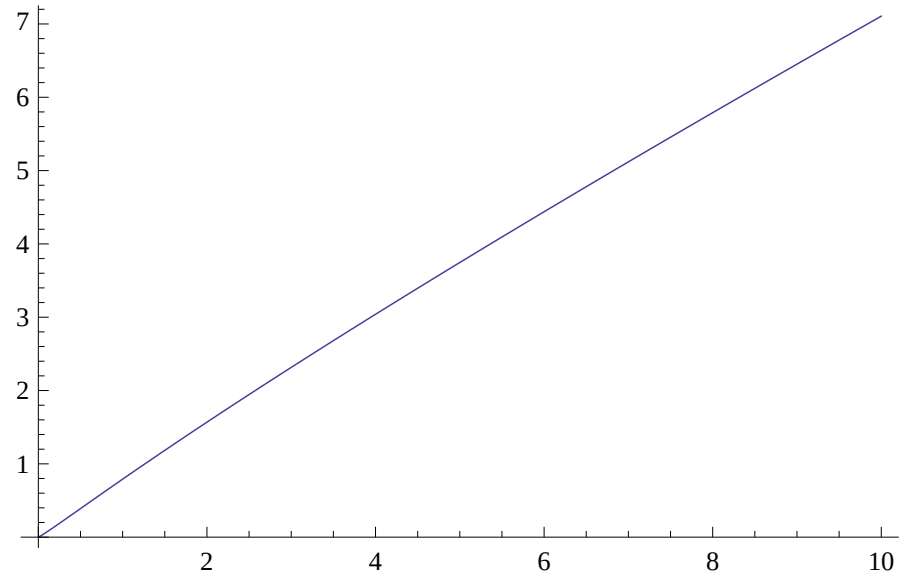

Figure 2: The biological death rate $r(t)$ of the first biological system

distribution function) and the fact that the only probability left in the expression is subject to the equality $\mathbf{P}\{\max(T_1 \ldots T_n) > t\} = (1 - \prod_{i=1}^{n} F_{T_i})$:

$$\begin{aligned}
\mathbf{P}\{\max(T_1, T_2, T_3) > T_4\} &= \int_0^\infty \mathbf{P}\{\max(T_1, T_2, T_3) > t\} f_{T_4}(t)\, \mathrm{d}t \\
&= \int_0^\infty \left(1 - F_{T_1}(t) F_{T_2}(t) F_{T_3}(t)\right) f_{T_4}(t)\, \mathrm{d}t \\
&= \int_0^\infty \left(1 - \left(F_W(t)\right)^3\right) f_e(t)\, \mathrm{d}t \\
&= \int_0^\infty \left(1 - \left(1 - e^{-\sqrt{t}}\right)^3\right) \frac{e^{-\frac{t}{2}}}{2}\, \mathrm{d}t
\end{aligned}$$

Again, using the notation that $F_W(t)$ is the cumulative distribution function of the Weibull$(1,\frac{1}{2})$ distribution, and $f_e(t)$ is the probability distribution function of the exp$(\frac{1}{2})$ distribution. This is also an ugly integral, best left in the capable hands of Mathematica:

```
In[7]:= Integrate[((1-(1-Exp[-Sqrt[t]])^3)*Exp[-t/2]/2),
         {t, 0, Infinity}]
Out[7]= (* Analytic and somewhat uninteresting answer *)
In[8]:= N[%]
Out[8]= 0.647406
```

Thus, the probability that component four is the component that causes biological death is approximately 65%. Hence, it would probably be of interest to add a (warm or cold) biologically redundant component parallel to that component.

1.3 Introducing redundancy

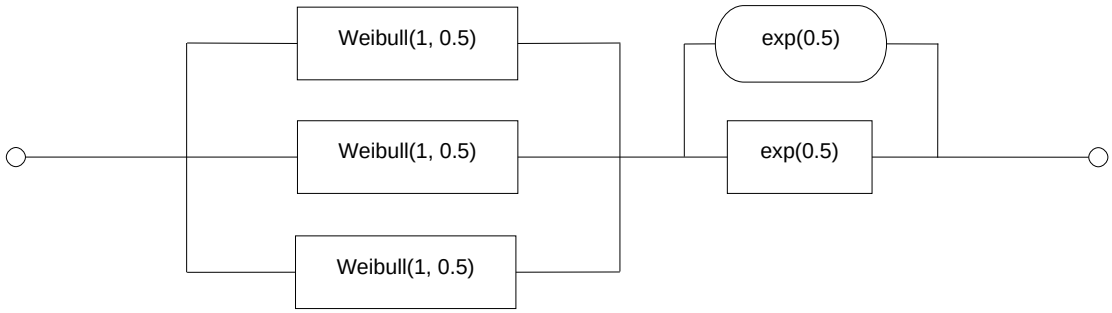

Figure 3: The system after the addition of a redundant component

To reduce the system's dependency on component four, one can couple it with a (warm or cold) biologically redundant component, perhaps of the exact same type (i.e. same distribution) as the existing one (see Figure 3). This is not difficult: all that is required is a modification of the biological survival function:

$$
\begin{aligned}
R_w(t) &= \left(1-(1-R_W(t))^3\right)\mathbf{P}\{\max(T_1,T_2)>t\} \\
&= \left(1-F_W(t)^3\right)\left(1-F_e(t)^2\right) \\
&= \left(1-\left(1-e^{-\sqrt{t}}\right)^3\right)\left(1-\left(1-e^{-\frac{t}{2}}\right)^2\right) \\
R_c(t) &= \left(1-(1-R_W(t))^3\right)\left(1-\int_0^\infty (1-R_{T_1}(t-x))\,R_{T_2}(x) r_{T_2}(x)\,\mathrm{d}x\right) \\
&= \left(1-F_W(t)^3\right)\left(\int_0^\infty F_e(t-x)\,(1-F_e(x))\,f_e(x)\,\mathrm{d}x\right) \\
&= \left(1-\left(1-e^{-\sqrt{t}}\right)^3\right)\left(1-\int_0^\infty \left(1-e^{-\frac{t-x}{2}}\right)e^{-\frac{x}{2}}\frac{e^{-\frac{x}{2}}}{2}\,\mathrm{d}x\right) \\
&= \left(1-\left(1-e^{-\sqrt{t}}\right)^3\right)\left(1-\frac{1}{2}\int_0^\infty e^{-x}-e^{-\frac{t+x}{2}}\,\mathrm{d}x\right) \\
&= \left(1-\left(1-e^{-\sqrt{t}}\right)^3\right)\left(\frac{1}{2}+e^{-\frac{t}{2}}\right)
\end{aligned}
$$

Here, T_1 and T_2 are the life lengths of the (now two) exponentially distributed components. Further, R_w represents the survival function of the system with a warm biologically reduntant component added, while R_c represents the system with a cold equivalent. Using these two definitions, we can define the appropriate death intensities r_w and r_c, to compare the two (hopefully) improved systems to the original one. Again, we'll let Mathematica do the dirty work for us:

```
In[9]:= Rw[t_] := (1-(1-Exp[-Sqrt[t]])^3)(1-(1-Exp[-t/2])^2)
```

```
In[10]:= Rc[t_] := (1-(1-Exp[-Sqrt[t]])^3)(1/2+Exp[-t/2])
In[11]:= rw[t_] := -D[Log[Rw[t]]]
In[12]:= rc[t_] := -D[Log[Rc[t]]]
In[13]:= Plot[{r[t], rw[t], rc[t]}, {t, 0, 10}]
Out[13]= -Graphics-
In[14]:= Export["fig2.eps", %]
Out[14]= fig2.eps
```

As seen in Figure 4, these modifications change the death rate significantly. The warm redundant component reduces death rate over the whole time period, although not significantly; the cold redundant component reduces death rate significantly, even achieving a negative death rate for $t < 1$. Clearly, the cold redundant component does a much better job than the warm one.

Frankly, this is not a surprise; if a component is being used, it is a risk of biological failure. Cold redundant components postpone usage until the latest possible moment, thus postponing component failure as much as possible. While this has a clear advantage, a warm redundant component is easier to "implement", since the biological system won't have to keep track of the state of the original component — it's a simple matter of starting everything at once.

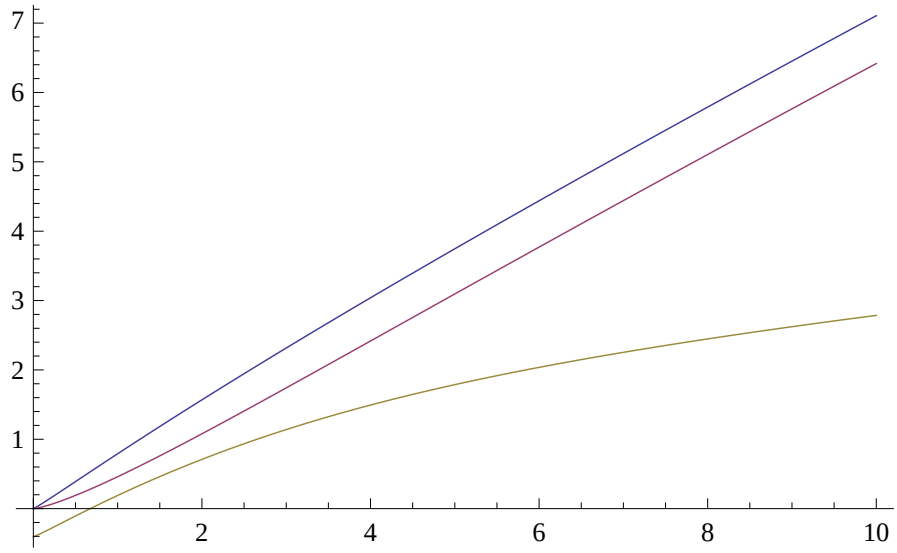

Figure 4: Biological death rates $r(t)$, $r_w(t)$ and $r_c(t)$

1.4 Improved biological components

Another possibility when it comes to both reducing death rates and increasing the expected life length is to replace failure-prone components with better ones. Here, we will experiment with replacing the fourth component with one of an $\exp(\rho)$ distribution, where $\rho < \frac{1}{2}$, and compare the results to both the cold and warm redundant component solution. A reduced parameter to the exponential distribution could be interpreted as a reduced failure rate, or an increase in mean value (and therefore life length of the component).

We start by calculating the expected life length using $R_w(t)$ and $R_c(t)$ (naming these E_w and E_c, respectively), so that we have values to compare to, and then set up a function $R(t, \rho)$ from which we can compute the expected life length depending on this new variable ρ, along with a corresponding function $E(\rho) = \mathbf{E}\{T|\rho\} = \int_0^\infty R(t, \rho)\, \mathrm{d}t$. We then plot $E(\rho)$ for $0 < \rho < \frac{1}{2}$:

```
In[15]:= Etw := Integrate[Rw[t], {t, 0, Infinity}]
In[16]:= Etc := Integrate[Rc[t], {t, 0, Infinity}]
In[17]:= R[t_, rho_] := Exp[-t*rho](1-(1-Exp[-Sqrt[t]])^3)
In[18]:= Erho[rho_] := Integrate[R[t, rho], {t, 0, Infinity},
                                 Assumptions -> rho >= 0]
In[19]:= Plot[{Erho[rho], Etw, Etc}, {rho, 0, 1/2}]
Out[19]= -Graphics-
In[20]:= Export["fig3.eps", %]
Out[20]= fig3.eps
```

As can be seen in Figure 5, the system's expected life length improves as ρ gets smaller — it is better than the warm redundant component for $\rho < 0.29$, while it doesn't outperform the cold redundant component until $\rho < 0.04$. This further illustrates the power of a cold redundant component; it is a *very* effective way to improve the expected life length of a system, given that it is coupled with the "worst" component.

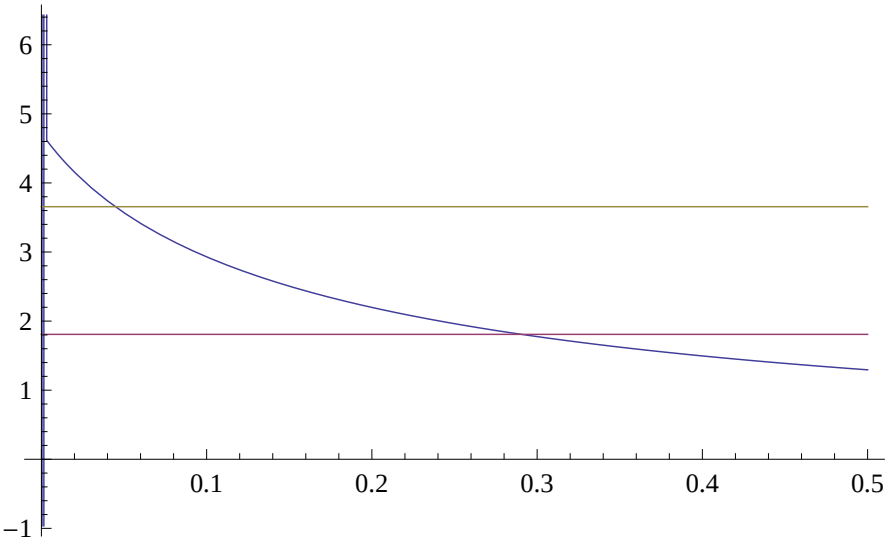

Figure 5: The expected life length $E(\rho)$ compared to the expected life length using a warm (E_w) and cold (E_c) redundant component

2 Introducing biological cost

It is obvious that one cannot choose biological components at will — more durable components are bound to be more difficult to produce or unattainable in some other way. To simulate this, we can introduce *biological cost*.

Consider the system in Figure 6, where μ and λ are variables related to the biological cost: each componens has an incurred cost of $1/5 + 1/\gamma$, where γ is the corresponding parameter of its biological distribution.

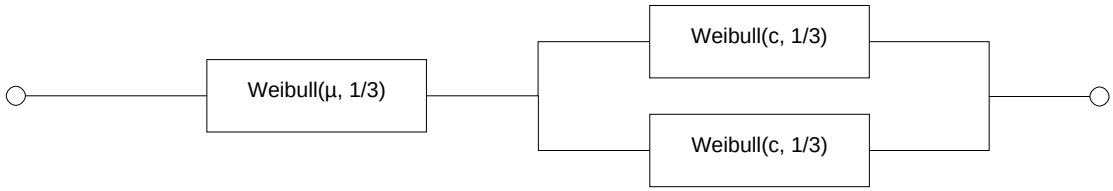

Figure 6: The variable biological system — note that c is referred to as λ in the text

Obviously, one wants to minimize biological cost while still maximizing the expected life length of the system. Since it is difficult to maximize one parameter while minimizing another, a better approach is to fix the total biological cost and maximize life length with respect to this. In order to to so, we first establish a relation between μ, λ and the total cost C:

$$C(\mu, \lambda) = \frac{3}{5} + \frac{1}{\mu} + \frac{2}{\lambda}$$

Next, we set up the expected life length $E_l(\mu, \lambda)$, dependent on the two parameters μ and λ. We start by setting up the biological survival function $R(t, \mu, \lambda)$, using it to express $E(\mu, \lambda)$:

$$\begin{aligned} R(t, \mu, \lambda) &= R_\mu(t) \left(1 - (1 - R_\lambda(t))^2\right) \\ &= (1 - F_\mu(t)) \left(1 - F_\lambda(t)^2\right) \\ &= e^{-(3t)^\mu} \left(1 - \left(1 - e^{-(3t)^\lambda}\right)^2\right) \end{aligned}$$

$$E_l(\mu, \lambda) = \mathbf{E}\{T|\mu, \lambda\} = \int_0^\infty R(t, \mu, \lambda)\ \mathrm{d}t = \int_0^\infty e^{-(3t)^\mu} \left(1 - \left(1 - e^{-(3t)^\lambda}\right)^2\right)\ \mathrm{d}t$$

In order to let Mathematica take care of these horrible things, we set up μ, $R(t, \mu, \lambda)$ and $E_l(\mu, \lambda)$ as Mathematica functions:

```
In[1]:= Cost[mu_,lambda_] := 3/5 + 1/mu + 2/lambda
In[2]:= R[t_,mu_,lambda_] := Exp[-(3t)^mu](1-(1-Exp[-(3t)^lambda])^2)
In[3]:= El[mu_,lambda_] := NIntegrate[R[t,mu,lambda], {t, 0, Infinity},
                            MaxRecursion->20, AccuracyGoal->8]
```

Our task is now to maximize $E_l(\mu, \lambda)$ for some different maximum values of C, say $C_{max} = 1, 2, \ldots 10$. For this, we can use Mathematica's `FindMaximum` function with the constraint that $C(\mu, \lambda) = C_{max}$, for example by generating a table containing the optimum values of μ and λ for each C_{max}:

```
In[4]:= results = Table[{m,l}/.Last[FindMaximum[{El[m,l],
                          Cost[m,l]==c,m>0,l>0},{m,l}]],{c,1,10}]
Out[4]= (* List of output values  *)
```

All that is left now is some kind of visualization of our data. We'll need to plot the life lengths to get a feel for the general trend, but we also need a table to properly display full data. We can generate the plot in Mathematica:

```
In[5]:= Els = El@@@results
Out[5]= (* List of output values *)
In[6]:= ListPlot[Els, Filling -> Axis, AxesOrigin -> {0,0}]
Out[6]= -Graphics-
In[7]:= Export["fig4.eps", %]
Out[7]= fig4.eps
```

The results (seen in Table 1 and Figure 7) indicate that an increased investment, i.e. increased maximum biological cost, will also result in an increased life length. The decline for $2 \leq C_{max} \leq 7$ does not contradict this; since we constrained the algoritm to only consider μ and λ satisfying $C(\mu, \lambda) = C_{max}$, we are in effect *requiring* that we spend all biological monetary units available to us — there is of course no harm in using less funds than we have, and we can therefore select the $C_{max} = 1$ solution for $2 \leq C_{max} \leq 7$ as well.

Table 1: Actual values of μ, λ and $E_l(\mu, \lambda)$ for different values of C_{tot}

C_{tot}	μ	λ	$E_l(\mu, \lambda)$
1	39.2745	5.2759	0.3146
2	42.0501	1.4532	0.2866
3	41.7723	0.8417	0.2687
4	36.2571	0.5930	0.2561
5	0.6863	0.6796	0.2269
6	0.5067	0.5838	0.2446
7	0.4123	0.5032	0.2758
8	0.3511	0.4394	0.3227
9	0.3073	0.3886	0.3900
10	0.2741	0.3477	0.4854

We can also note that μ and λ, while both decreasing (almost) monotonically, don't decrease in proportion to each other. For small values of C_{max}, μ is *much* larger than λ — this indicates that decreasing λ has a bigger effect than decreasing μ, which is counter-intuitive. When C_{max} gets larger, the reductions of λ start pushing the limit, and reducing μ becomes a more feasible solution; when the life length starts increasing again μ is smaller than λ. This makes sense as the single, uncoupled component is the one that has the largest impact in situations when all component are so similarly distributed as in this system.

The result is to be expected; since an increased cost implies a smaller value of μ and/or λ, it also implies (due to how the Weibull distribution behaves) that the failure rate increases less over time. In fact, for values of these parameters less than one, the failure rate *decreases* over time; for values larger than one the rate *increases*.

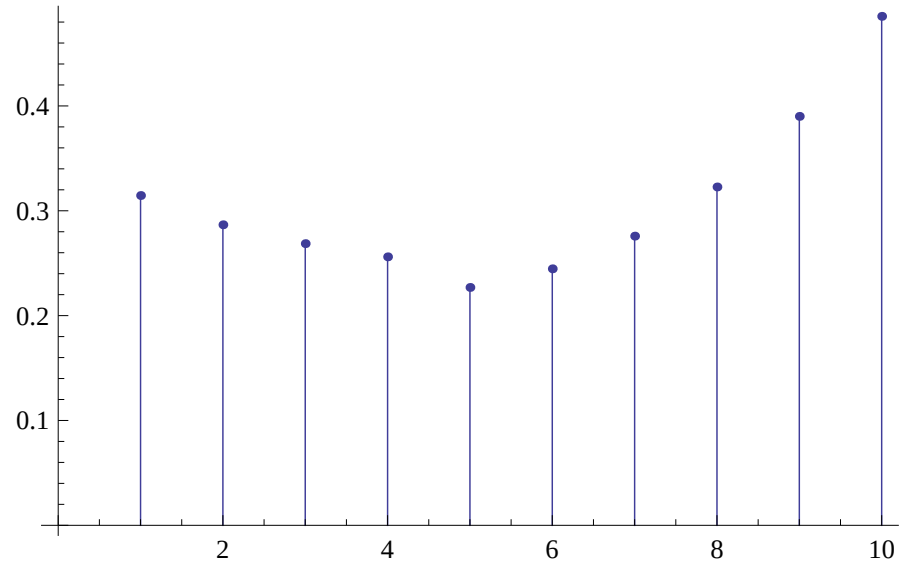

Figure 7: The expected life length of the optimal system with increasing biological cost

2.1 Technical difficulties

One should also comment on the optimization process itself; since it consists of a large number of numerical integrations (one for each evaluation `FindMaximum` does) omptimizing $E_l(\mu, \lambda)$ is both computationally unstable, and in some cases unstable.

To get any sensible results, the integration method had to be instructed to double its maximum recursion level and its accuracy goal to remedy the issues that occur when the integration result is very close to 0. This is because Mathematica's `NIntegrate` function uses relative tolerance as a measure of accuracy, making integrals close to 0 difficult to compute.

Of course, this has a downside; the result is (possibly much) less accurate. This, along with the fact that `FindMaximum` only aims to find local maxima, as opposed to `NMaximize` that finds global maxima, makes the result somewhat unreliable.