The Vector Mismatches paper model analyses

*Samraat Pawar (mhasoba@gmail.com)

In [4]:

```
#Load some modules etc
%matplotlib inline
import matplotlib.pyplot as plt
from sympy import *
import scipy as sc
import numpy
init_printing()
```

1. The Euler-Lotka equation-based model for r

*To dos:

1. Find approximation using Gompertz function for mortality rate part of l_x, that is, mortality rate = z exp(b t) instead of just z

Assign symbolic functions:

In [5]:

```
x, b_pk, v, a, z, z_J, kappa, T, M0, K, r, t = var('x b_pk v a z z_J kappa T M0
 K r t')

la = exp(-z_J*a)

lx = la * exp(-z*(x - a)); simplify(lx)
```

Out[5]:

$$e^{-az_J+z(a-x)}$$

In [6]:

```
# alternative:
# bx = b_pk * exp((kappa-1)*(x-v-a)/-v) * (x-a)**(kappa-1) * v**(1-kappa) # *Not
e that kappa > 1, always!
# also consider the shifted gompertz
# bx = b_pk * exp(-kappa*(x-v-a)); simplify(bx) # with delay till peak fecundity
 v

bx = b_pk * exp(-kappa*(x - a)); simplify(bx)
```

Out[6]:

$$b_{pk}e^{\kappa(a-x)}$$

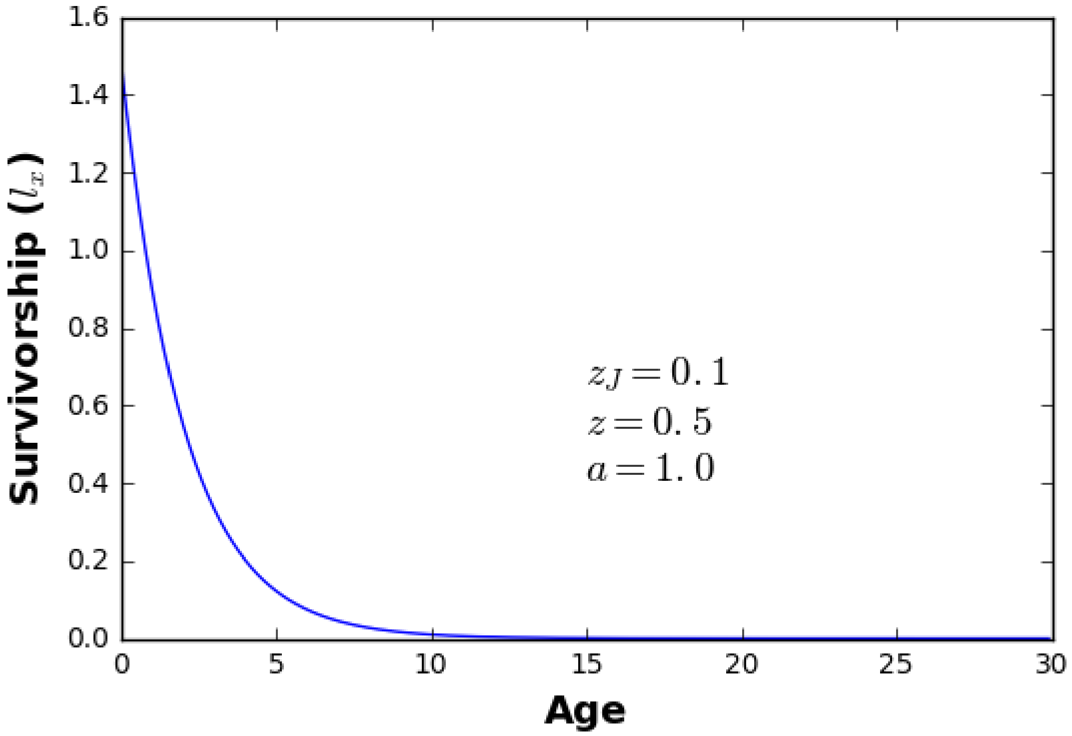
Check the functions

In [7]:

```
#assign some parameter values
z_J_par = .1
z_par = .5
a_par = 1.
b_pk_par = 1.
# v_par = 1
kap_par = .5
```

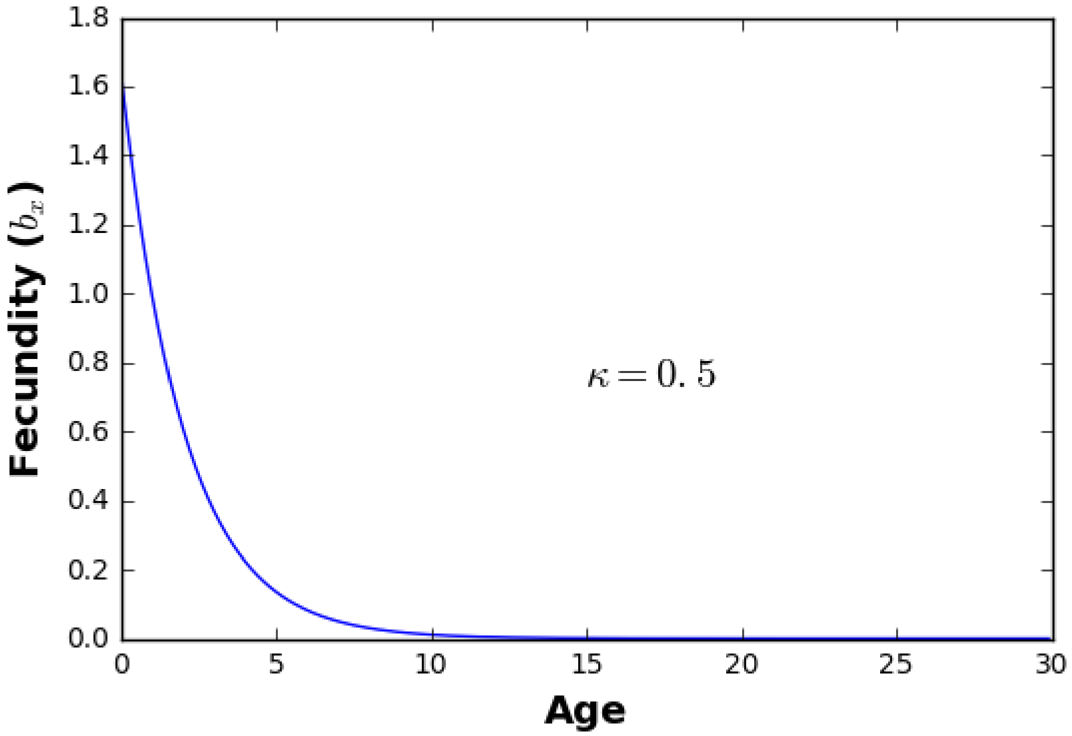
In [8]:

```
#Now numerically evaluate
x_vec = numpy.arange(0, 30, 0.1) #vector of ages
lx_vec = numpy.array([lx.evalf(subs = {z_J:z_J_par, x:age, a:a_par, z:z_par}) for age in x_vec])#use lambidy to speed up

fig = plt.figure(); ax = fig.add_subplot(111)
ax.plot(x_vec, lx_vec);
ax.set_xlabel('Age', fontsize=14, fontweight = 'bold');
ax.set_ylabel('Survivorship ($l_x$)', fontsize=14, fontweight = 'bold')
ax.text(sc.mean(x_vec), sc.amax(lx_vec)/2,
        '$z_J = ' + str(z_J_par)+'$ \n' +
        '$z = ' + str(z_par)+'$ \n' +
        '$a = ' + str(a_par)+'$ \n',
        horizontalalignment='left', verticalalignment='top', fontsize=15)
plt.savefig('../results/lxModel.pdf')
```



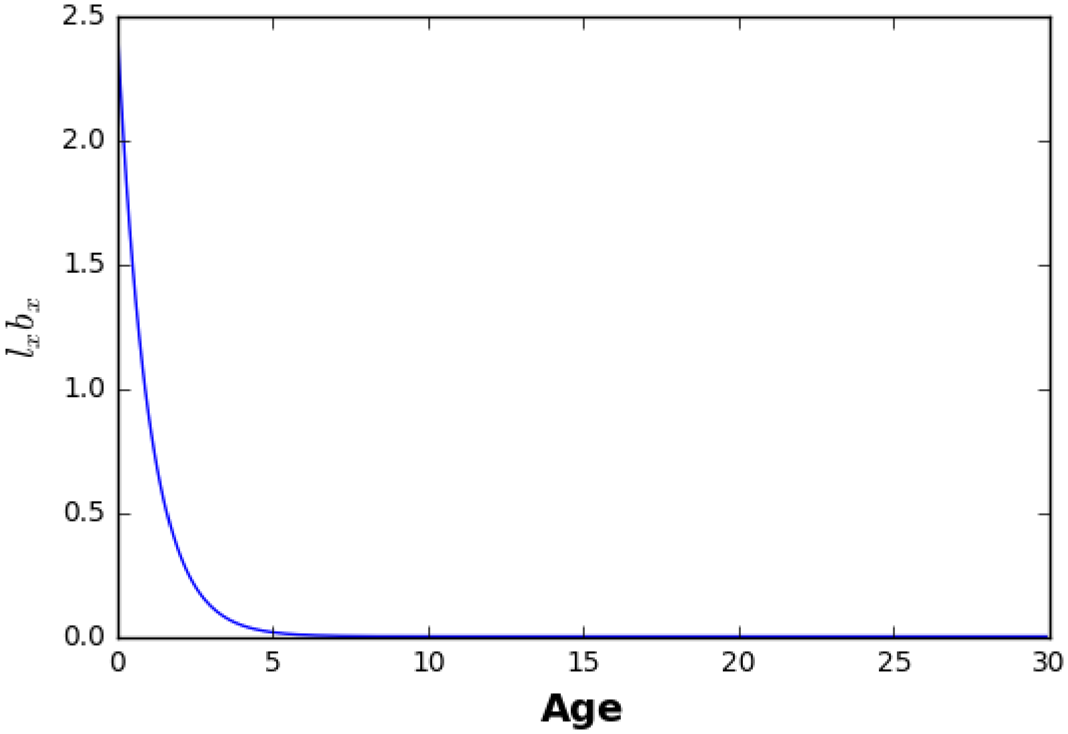
In [9]:

```
fig = plt.figure(); ax = fig.add_subplot(111)
bx_vec = numpy.array([bx.evalf(subs = {b_pk:1, x:age, a:a_par, kappa:kap_par}) f
or age in x_vec])#use lambidy to speed up

ax.plot(x_vec, bx_vec);
ax.set_xlabel('Age', fontsize=14, fontweight = 'bold');
ax.set_ylabel('Fecundity ($b_x$)', fontsize=14, fontweight = 'bold')
ax.text(sc.mean(x_vec), sc.amax(bx_vec)/2,
        '$\kappa = ' + str(kap_par)+'$ \n',
        horizontalalignment='left', verticalalignment='top', fontsize=15)
plt.savefig('../results/bxModel.pdf')
```



In [10]:

```
fig = plt.figure(); ax = fig.add_subplot(111) #now plot lxmx

ax.plot(x_vec, bx_vec*lx_vec);
ax.set_xlabel('Age', fontsize=14, fontweight = 'bold');
ax.set_ylabel('$l_x b_x$', fontsize=14, fontweight = 'bold')
# ax.text(sc.mean(x_vec), sc.amax(bx_vec)/2,
#         '$v = ' + str(v_par)+'$ \n' +
#      z_J   '$\kappa = ' + str(kap_par)+'$ \n',
#         horizontalalignment='left', verticalalignment='top', fontsize=15)
plt.savefig('../results/lxbxModel.pdf')
```



In [11]:

```
EuLo = exp(-r * x) * lx * bx; simplify(EuLo)
```

Out[11]:

$$b_{pk}e^{-az_J+\kappa(a-x)-rx+z(a-x)}$$

In [12]:

```
# integrate(EuLo,(x,a,oo)) #doesn't work - parameters need to be constrained
# Integrate EuLo on sagemath using integrate(EuLo, x, a, infinity), with positiv
ity constraints on all parameters
EuLo_int = b_pk*exp(-a*z_J)/((kappa + r)*exp(a*r) + z*exp(a*r)); simplify(EuLo_i
nt)
```

Out[12]:

$$\frac{b_{pk}e^{-a(r+z_J)}}{\kappa+r+z}$$

In [13]:

```
#the exact solution
r_SP = solve(EuLo_int-1,r); r_SP = simplify(r_SP[0])
type(r_SP);r_SP
```

Out[13]:

$$\frac{1}{a}\left(-a\left(\kappa+z\right)+\mathrm{LambertW}\left(ab_{pk}e^{a\left(\kappa+z-z_J\right)}\right)\right)$$

In [14]:

```
#the approximation
series(ln(r + kappa + z),r, 0,2) #approximate just the offending term after taki
ng log
# exp(log(kappa + z) + r/(kappa + z))
```

Out[14]:

$$\log\left(\kappa+z\right)+\frac{r}{\kappa+z}+\mathcal{O}\left(r^2\right)$$

In [15]:

```
#substitute # exp(log(kappa + z) + r/(kappa + z))
EuLo_int_app = b_pk*exp(-a*z_J)/ exp(log(kappa + z) + r/(kappa + z))

r_SP_app = solve(EuLo_int_app-1, r); r_SP_app = simplify(r_SP_app[0]); r_SP_app
```

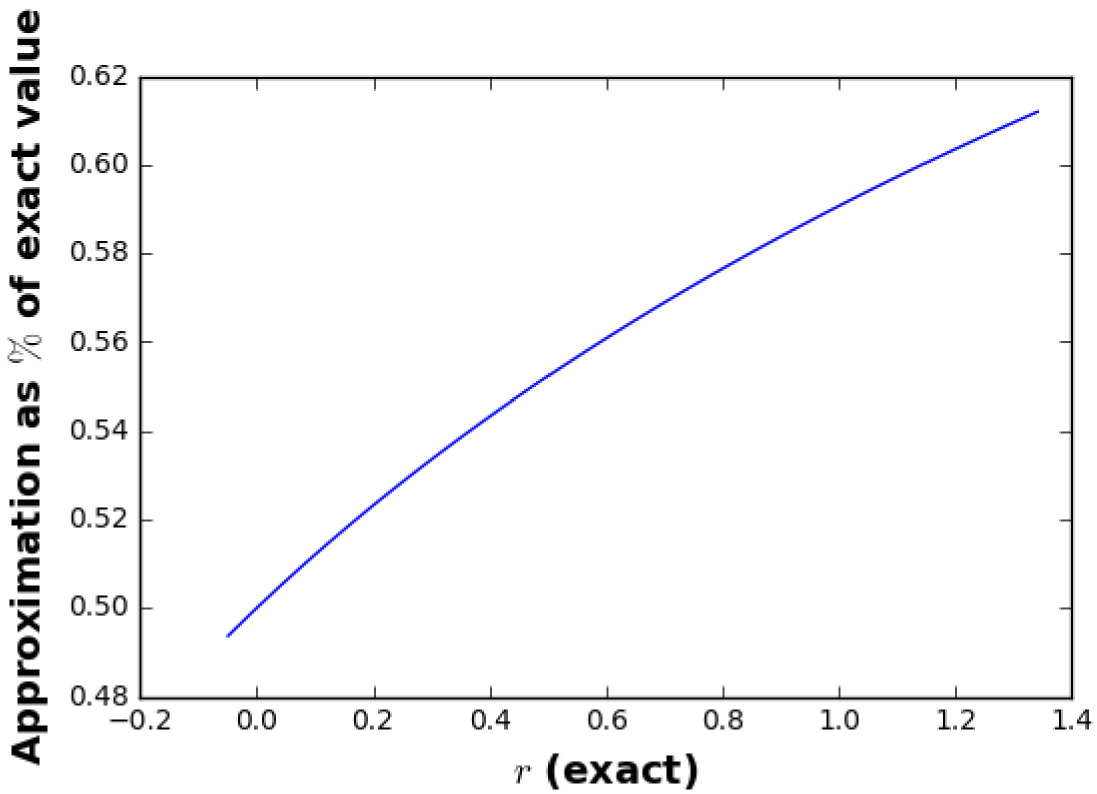
Out[15]:

$$\left(\kappa+z\right)\log\left(\frac{b_{pk}e^{-az_J}}{\kappa+z}\right)$$

Check the exact vs approx solutions for r

In [16]:

```
from scipy.special import lambertw # need this for the lambertw

b_pk_vec = numpy.arange(1, 10, 0.1).astype(float) #vector of b_pk's
r_SP_app_vec = numpy.array([r_SP_app.evalf(subs = {b_pk:b_pkVal,z_J:z_J_par, a:a
_par, z:z_par, kappa:kap_par}) for b_pkVal in b_pk_vec])#use lambidy to speed up

# have to specify r_SP with lambertw:
# r_SP_vec = (-a*(kappa + z) + lambertw((a_par*b_pk_vec*exp(a_par*(kap_par + z_p
ar - z_J_par))).astype('float')).real)/a_par
tmp = lambertw((a_par*b_pk_vec*exp(a_par*(kap_par + z_par - z_J_par))).astype('f
loat')).real
r_SP_vec = -(kap_par + z_par) + tmp/a_par

# -(kap_par + z_par) + lambertw((a_par*b_pk_vec*exp(a_par*(kap_par + z_par - z_J
_par))).astype('float')).astype('float')/a_par # specify r_SP correctly with lam
bertw

fig = plt.figure(); ax = fig.add_subplot(111)
ax.plot(r_SP_vec, r_SP_vec/r_SP_app_vec)
ax.set_xlabel('$r$ (exact)', fontsize=14, fontweight = 'bold');
ax.set_ylabel('Approximation as $\%$ of exact value', fontsize=14, fontweight =
'bold')

# ax.plot(b_pk_vec, r_SP_app_vec);
# ax.set_xlabel('$b_{pk}$', fontsize=14, fontweight = 'bold');
# ax.set_ylabel('$r$ (approximation)', fontsize=14, fontweight = 'bold')

# ax.text(sc.mean(r_vec), sc.amax(sol_vec)/2,
#         '$z_J = ' + str(z_J_par)+'$ \n' +
#         '$z = ' + str(z_par)+'$ \n' +
#         '$a = ' + str(a_par)+'$ \n',
#         horizontalalignment='left', verticalalignment='top', fontsize=15)
plt.savefig('../results/rapprox.pdf')
```



*r can be translated into population density using the logistic growth equation, which has the solution:

In [27]:

```
M = M0*K*exp(r * t)/(K + M0*(exp(r * t) -1)); M
```

Out[27]:

$$\frac{K M_0 e^{rt}}{K + M_0 \left(e^{rt} - 1\right)}$$

2. Temperature-dependence of life history parameters

Assign symbolic functions

In [19]:

```
B_0, E_A, E_D, T_pk, k = symbols('B_0 E_A E_D T_pk k')

B = B_0 * exp(-E_A/(k * T)) / (1 + (E_A / (E_D - E_A)) + exp((E_D / k)*
((1/T_pk) - (1/T)))); B
```

Out[19]:

$$\frac{B_0 e^{-\frac{E_A}{Tk}}}{\frac{E_A}{-E_A+E_D} + e^{\frac{E_D}{k}\left(\frac{1}{T_{pk}} - \frac{1}{T}\right)} + 1}$$

Assign some parameter values

In [20]:

```
k = 8.617 * 10**-5
E_A_par = 1.
E_D_par = 4.
T_pk_par = 25.

b0_bpk = 0.9
b0_v_in = 0.012
b0_a_in = 0.0045
b0_z_in = 2.5
b0_z_J_in = 1.9
k_cconstant = 2
```

3. The temperature-dependent sensitivity analysis

*Note: You can also do M.subs(r,r_SP) to get the full expression for M with r approximation substituted

Now calcuate derivatives

In [33]:

```
test1 = simplify(diff(M.subs(r,r_SP_app), b_pk)) #nasty!
test1
```

Out[33]:

$$\frac{KM_0te^{t(\kappa+z)\log\left(\frac{b_{pk}e^{-az_J}}{\kappa+z}\right)}}{b_{pk}\left(K+M_0\left(e^{t(\kappa+z)\log\left(\frac{b_{pk}e^{-az_J}}{\kappa+z}\right)}-1\right)\right)^2}(\kappa+z)\left(K+M_0\left(e^{t(\kappa+z)\,\mathrm{l}}\right.\right.$$

In [30]:

```
simplify(diff(M, r))
```

Out[30]:

$$\frac{KM_0te^{rt}}{\left(K+M_0\left(e^{rt}-1\right)\right)^2}\left(K+M_0\left(e^{rt}-1\right)-M_0e^{rt}\right)$$