THE INFLUENCE OF POPULATION STRUCTURE ON GENETIC VARIATION IN CAPTIVE BRED SPECIES

A Thesis
Presented to
The Graduate Faculty of The University of Akron

In Partial Fulfillment
Of the Requirements for the Degree
Master of Science

Andrew Treymane Jones
December, 2014
THE INFLUENCE OF POPULATION STRUCTURE ON GENETIC VARIATION IN CAPTIVE BRED SPECIES

Andrew Treymane Jones

Thesis

Approved:

Advisor
Dr. Francisco B. G. Moore

Accepted:

Dean of the College
Dr. Chand Midha

Committee Member
Dr. Matthew Shawkey

Dean of the Graduate School
Dr. George Newkome

Committee Member
Dr. Randall Mitchell

Date

Committee Member
Dr. Zhong-Hui Duan

Department Chair
Dr. Monte Turner

ii
ABSTRACT

As humans continue to influence ecosystems, the preservation of endangered species becomes increasingly important. This leads to the use of captive breeding programs as a tool while fragile habitats are restored or recover. However, the majority of captive breeding studies deal with parameters that account for optimization only while in captivity. Breeding plans look to increase fitness and or fecundity while minimizing deleterious effects associated with inbreeding and small population sizes while in captivity. Few studies focus on the interaction between inbreeding and fitness in captivity with success when released into the wild. Fewer studies still have examined the utilization of genetic variation that is partitioned within and between captive populations in meeting the objectives of breeding plans. Largely missing from current breeding plans is recognition of potential tradeoffs between fitness in the captive environment and adaptability to release environments. We simulated how population structure in and out of captivity influences long-term species success. We specifically tested if a traditional breeding design (maximum avoidance of inbreeding) preformed as well as a model that maintained traditional segregation of alleles in captivity. We found that maintaining inbred wild population structure protected genetic variation better than maximum outcrossing while in captivity. This study calls into question current practices in captive breeding when future release of captive bred populations is anticipated.
ACKNOWLEDGEMENTS

This work was funded primarily by the National Science Foundation (NSF grant # 0844198), and the Choose Ohio First Bioinformatics Grant. I would like to thank Paco Moore for his guidance and hours of academic banter. I would like to acknowledge Dr. Duan and Dr. Londraville for their work in getting additional funding at the University of Akron for work like this. I would like to thank Alissa Calabrese and Katherine Campbell for their tremendous support and help with this manuscript. I would like to thank my committee for their patience and my lab mates for their continued support (Anne Hall, Ashley Bair, Ashley Wain, Chris Marks, Ramsey Langford, K.B. Nguyen, and Sylvia Jones).
# TABLE OF CONTENTS

<table>
<thead>
<tr>
<th>LIST OF FIGURES</th>
<th>vi</th>
</tr>
</thead>
<tbody>
<tr>
<td>CHAPTER</td>
<td></td>
</tr>
<tr>
<td>I. INTRODUCTION</td>
<td>1</td>
</tr>
<tr>
<td>II. MATERIALS AND METHODS</td>
<td>7</td>
</tr>
<tr>
<td>III. RESULTS</td>
<td>14</td>
</tr>
<tr>
<td>IV. DISCUSSION</td>
<td>21</td>
</tr>
<tr>
<td>V. SUMMARY AND CONCLUSION</td>
<td>26</td>
</tr>
<tr>
<td>REFERENCES</td>
<td>29</td>
</tr>
<tr>
<td>APPENDICES</td>
<td>32</td>
</tr>
<tr>
<td>APPENDIX A. SOURCE CODE FOR SEED POPULATIONS</td>
<td>33</td>
</tr>
<tr>
<td>APPENDIX B. SOURCE CODE FOR CAPTIVE RUNS</td>
<td>60</td>
</tr>
<tr>
<td>APPENDIX C. PARAMETER FILE FOR SEED POPULATIONS</td>
<td>93</td>
</tr>
<tr>
<td>APPENDIX D. PARAMETER FILE FOR CAPTIVE RUNS</td>
<td>98</td>
</tr>
</tbody>
</table>
# LIST OF FIGURES

<table>
<thead>
<tr>
<th>Figure</th>
<th>Description</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>Symmetrical 2 Locus 2 Allele Additive Model Under Stabilizing Selection Showing Genetic Redundancy</td>
<td>5</td>
</tr>
<tr>
<td>2</td>
<td>Fitness Surface for Wild and Captive Populations</td>
<td>12</td>
</tr>
<tr>
<td>3</td>
<td>Fitness for MAI and DS treatments during captivity</td>
<td>14</td>
</tr>
<tr>
<td>4</td>
<td>Average $F_{is}$ for DS and MAI populations during captivity</td>
<td>15</td>
</tr>
<tr>
<td>5</td>
<td>Average $F_{st}$ measures for DS and MAI populations in captivity</td>
<td>17</td>
</tr>
<tr>
<td>6</td>
<td>Average fitness measures for DS and MAI populations after release</td>
<td>18</td>
</tr>
<tr>
<td>7</td>
<td>Average $F_{st}$ for DS and MAI populations after release</td>
<td>19</td>
</tr>
<tr>
<td>8</td>
<td>Census for DS and MAI populations after release</td>
<td>20</td>
</tr>
</tbody>
</table>
CHAPTER I
INTRODUCTION

Optimizing captive genetic variation for long-term species success is both a complex as well as increasingly pertinent topic in conservation studies. Maintenance of genetic variation and the subsequent adaptability of the species are important considerations in developing successful captive breeding strategies (Frankham 1995, 2006; Rudnick and Lacy 2008, Meffert et al. 2005). However, captive environments and or zoos provide novel selective forces, and that selection tends to reduce the variation that can be used for adaptability (Frankham 2006). Genetic variation can exist within individuals, between individuals or between different breeding populations. In this study, we explore two breeding plans that stress maintenance of genetic variation at the opposite poles of those two levels. We utilize computational simulations to investigate the degree to which population structure is an important agent in determining captive fitness verses success upon release.

The loss of genetic variation via inbreeding, often due to population size, has long been thought to be one of the greatest threats to captive species (Lande 1988). The avoidance of the stochastic loss of alleles (decreased genotypic variance) and inbreeding depression have made Maximum Avoidance of Inbreeding (MAI) (Kimura and Crow 1963) the preferred strategy for most captive breeders (Meffert et al. 2005, Frankham 2006). In an ideal scenario, MAI programs would cross the most distantly related parents
in each generation to minimize the increased homozygosity due to shared ancestry. MAI strategies often minimize inbreeding by tracking ancestry via studbooks and outcrossing as much as possible within financial and geographic constraints. These regimes can increase the fecundity of the captive individuals (Frankham 2006). The very sound logic here is that inbreeding depression can decrease fecundity in the captive environment, potentially jeopardizing survival of captive populations. For this reason the populations are continuously outcrossed to the maximum extent possible.

Alternatively, captive breeding programs could utilize a Selective Purging (SP) plan. Inbreeding in conjunction with selection can result in the purging of deleterious alleles (Crnokrak and Barrett 2002). In Speke’s gazelle, this mode of breeding was attempted with novel success (Templeton and Read 1984). However, the use of SP in captive breeding is rare, and in one test, SP performed poorly relative to MAI (Meffert et al. 2005). Reduced fecundity and decreased survivorship while in captivity certainly make SP a risky strategy to utilize. However, population dynamics before, during, and after breeding programs may prove to be vital in determining which type of strategy is best for preserving the population as well as its genetic diversity.

MAI and SP are designs for captive breeding that can influence the adaptive potential for released populations. Outcrossing as mandated by MAI creates a single large breeding population. In a single large population, adaptation to a captive environment necessarily results in a decrease in genetic variation in as little as a single generation (Briscoe et al. 1992, Gilligan and Frankham 2003, Frankham 2008, and Christie et al. 2012). Genetic variation that is eliminated by natural selection for a captive environment is variation that is not available for selection to a novel wild environment upon release.
Alternatively, if a captive population is segregated into multiple small populations, selection still acts. If variation exists between populations, that proportion cannot be removed by selection. As subpopulation size decreases, heterozygosity is lost (Hartl 2000) exposing variation resulting from recessive alleles. This exposure of recessive alleles to increased selection is the basis for an SP strategy (Templeton and Read 1984).

The captive breeder must optimize population parameters to slow genetic loss via selection and drift while allowing for the highest level of adaptability when populations are released (Frankham 2006). Populations of as many as 100 individuals can fix up to 20% of their allelic variation in 100 generations, even in ideal circumstances (Chesser et al. 1980). An MAI strategy during captive breeding where all subdivided captive populations effectively serve as one large population slow but do not stop fixation of alleles stochastically and through selection to the captive environment. An SP strategy if implemented in a single population will purge recessive deleterious alleles when present but without multiple populations involved it will also experience loss of alleles due to selection for the captive environment and stochasticity. If multiple locally breeding subpopulations are used to create inbred demes then selection will still tend to purge deleterious recessive alleles, but selection will act independently in each deme preventing loss of variation that exists between demes. This type of Deme Structured (DS) breeding plan allows selection to act on genetic variation between individuals (purging) but preserves genetic variation between demes (theory reviewed in Chesser et al. 1980).

Goldstein and Holsinger (1992) demonstrate that structured populations can maintain variation through genetic redundancy and stabilizing selection. Genetic redundancy is a form of epistasis created whenever several different allelic combinations
can achieve a single phenotypic optimum. This can be demonstrated in a purely phenotypically additive two-locus two-allele system without dominance (Figure 1, see Whitlock et al. 1995). Under stabilizing selection the intermediate trait value can occur two different ways: 1) both loci are heterozygous 2) one locus is homozygous for increased trait value and the other is fixed for decreased trait value. There is only one combination of alleles for which an individual in this population can have the maximum or minimum trait value. But there are six allelic combinations (four shown in figure 1) in which an individual in this population can get the maximum phenotypic value. Genetic redundancy creates a circumstance where, even in a purely adaptive trait, local populations can differ in their response to stabilizing selection. In cases where genetic redundancy exists we expect that variation to be maintained between demes despite uniform selection (Goldstein and Holsinger 1992). In a deme structured population the Whalund effect will generate variation between demes in allele frequency, alleles will be lost, and homozygosity will increase via drift. Under stabilizing selection any homozygous combinations for a phenotypic optimum can be preserved due to reduced efficacy of selection.
Figure 1. Symmetrical 2 Locus 2 Allele Additive Model Under Stabilizing Selection Showing Genetic Redundancy. The maximum trait value occurs when there are four dominant alleles (AABB), two at locus A and B. The minimum trait value occurs when there are no dominant alleles at locus A or B (aabb). The optimum phenotype occurs when there is any combination of two dominant alleles between the A and B loci. Circled are the two homozygous variants that display the optimal phenotype. (See Goldstein and Holsinger 1992)

While the emphasis of most breeding plans such as MIA or SP is on maintaining a viable captive program, an increasing emphasis of breeding programs is to prepare a population for release into the wild. We propose a change in strategic viewpoint in these cases to exchange captive viability for adaptive viability of released populations. We suggest that optimizing interdemic variation at the expense of captive fitness does exactly that. MAI optimizes captive fitness via perpetual outcrossing, but exposes populations to captive selective forces that may fix populations for a phenotype that is inappropriate upon release. What is needed are tests of the relative performance of carefully optimized outcrossing and inbreeding programs that inform managers of the relative merits of different strategies both in captivity and after release.
In this study, we simulate an MAI and a DS strategy including both inbreeding effects and stabilizing selection. We then examine the influence of these strategies in stable captivity and after release into variable environments to measure recovery rates after release from captivity. We examine if either strategy is universally beneficial in and out of captivity. We follow measures of genetic variation (inbreeding coefficients and phenotypic variation) across the phases of the captive breeding program. Finally we compare the two breeding programs in terms of long-term survivorship.
CHAPTER II
MATERIALS AND METHODS

Simulation Overview

Individual based Monte Carlo simulations were used to test the relative efficacy of two breeding programs. Paired simulations of MAI and DS style breeding plans used identical individuals drawn from a ‘wild’ population. Once drawn from that wild population, captive bred populations were held in captivity for 20 generations before release into a new wild environment (Frankham 2006). Paired simulations also shared identical release environments.

A FORTRAN program, originally written (Moore and Tonsor 1994) to examine adaptive dynamics in a deme-structured environment was modified to test the utility of these two breeding plans in producing captive populations that are successful upon release from captivity. The simulations allowed for selective changes between wild and captive environments, drift mediated changes in gene frequencies and inbreeding depression due to increases in deleterious recessive homozygotes. Inbreeding load, response to stabilizing selection across environments, and demographic success were all tracked across generations. The purpose of these simulations was to assess animal management protocols under the different breeding regimes.
Simulated Organisms and Populations

All populations were comprised of diploid, obligately sexual, semelparous organisms with two separate sexes. All loci had a mutation rate ($\mu$) of $1 \times 10^{-5}$ for both forward and back mutations (Lynch 2010). Deme sizes both in and out of captivity were constrained by limits on reproduction that occurred in a density dependent manner. In both natural and captive demes populations were limited by resources to an upper limit (K) on the number of individuals that could stably be supported in a given environment. Population growth rates ($r$) tend to force the population at any time (t) toward K following the logistic (Verhulst 1838) function,

$$N_{(t+1)} = N_{(t)} + r N_{(t)}(K - N_{(t)})/K.$$ 

That function determined the number of individuals that would recruit into a deme from the previous generation such that a population recovering from low numbers would not necessarily recover within a single generation. A constant $r$ of 1.1 was used in all simulations. This value simulates a species that is moderately slow in its intrinsic growth rate as might be expected for many species requiring captive breeding as an ecological intervention.

Migration between wild demes was controlled by the function with the gamma distribution,

$$\text{Prob}(X) = m^x(1-m)$$

where $X$ is the number of demes away that a migrant would be moving and $m$ is the per capita migration rate (Moore and Tonsor 1994). In wild populations individuals were allowed to move up to 5 demes from their natal deme after the juvenile stage and before
the reproductive stage. Individuals migrating were randomly placed in a deme at the chosen distance. If placed in a deme where no other individuals were present these individuals were incapable of breeding as would be the case in a real population.

*Juvenile Mortality*

Density independent mortality during the juvenile period of the simulated organism occurred due to stabilizing selection on a Quantitative trait, inbreeding load, and stochastic mortality.

*Stabilizing Selection on a Quantitative Trait*

Fitness was partially determined by a function that allowed for a simple model that included a single phenotypic optimum with genetic redundancy (Goldstein and Holsinger 1992). The function is described by the equation,

$$P = 1 - |s_q (q_p - q_v)|$$

where $P$ represents the probability for survival as determined by the selection gradient ($s_q$), which is the difference between the optimal and lowest fitness for the quantitative trait, the optimum quantitative trait value ($q_p$), and the quantitative trait value of the individual ($q_v$). For instances when $q_p$ was equal to $q_v$, $P$ was set to a value of 1 to avoid division by 0. This created a survival decrease as individuals diverged from the phenotypic optimum in either direction. (See figure 2 for an example.) In this example the intermediate phenotypic value of 16, individuals with allelic values of 0 or 32 would
experience the full effect of the fitness differential. Individuals with allelic values at 8 or 24 would only experience half of s and demonstrate an overall higher fitness value. The actual location of the optimum depended on the particular simulation being run.

The quantitative phenotype for any individual was determined by a 16 locus, 2 allele genetic model. Each locus contributed equally to the trait with alleles contributing either -1, or a +1 to the final trait. The contributions of the two alleles at each locus were summed across all 16 loci in an individual to determine the trait value for that individual.

*Inbreeding Load*

Additionally, 100 loci were included in the model to simulate inbreeding depression caused by the build up of deleterious recessive alleles (see equation below). Inbreeding loci that were homozygous for the recessive alleles demonstrated relative fitness decrements of 0.02 per locus.

*Stochastic Mortality*

Mortality due to effects of factors not associated with the quantitative genetic trait for fitness can stochastically influence allele frequencies for this focal trait as well as deme sizes. This will add to the drift of alleles in both wild and captive populations. In order to reflect the contribution of stochastic mortality (L) to drift and demographic processes along with the influence of inbreeding was incorporated into the final survival probability (W) function as follows,

\[
W = (1-L) - \frac{s_i}{(q_p-q_v)} - (jq_i)
\]

where j is the number of inbreeding loci that were homozygous recessive and s_i is the
selection differential for inbreeding loci which was set to .02. L was held constant at 0.10 and $s_q$ was held constant at 0.4 throughout all portions of the simulations (Moore and Tonsor 1994).

Phase I: Founding Populations

In all simulations 100 individuals were drawn from a wild population that had been generated in the following way. At the outset 100 individuals were seeded into 100 demes. All individual genotypes were then initialized to one extreme of the fitness surface, (homozygous for every locus) and reared 5000 generations at which time mutation selection balance was evident at all loci. Carrying capacity (K) in wild populations was set to 100. The migration rate (m) was set to 0.005. The phenotypic optimum was set at 16. The difference between the phenotypic optimum and the phenotypic minimum (the selection gradient, $s$) was set to 0.4, with a 10% chance of stochastic death (L) during ontogeny.

Phase II: Captive Populations

To seed the captive regimes, a random deme from Phase I was selected and 10 individuals from that deme were chosen preserving the demic structure of the original environment. This was repeated to make 40 seed populations (20 sets of 2 identical seed populations) which each underwent the two breeding regimes in a paired design (breeding regimes are further detailed below). The optimum trait value was 24 in
captivity. This simulated a captive environment that differed from the original environment thus allowing adaptation during captivity. Animals were reared for 20 generations and tracked for 80 generations after release into novel environments.

*Deme Structured*

For Deme Structured (DS) regimes, no migration was allowed during captivity. Captive populations were limited to a carrying capacity of 10, the phenotypic optimum was set to 24, and there were 10 demes total in the population. All other factors remained consistent with the initial wild population in the simulation.

Figure 2. Fitness Surface for Wild and Captive Populations. Optimal trait value for initial wild populations was set to 16 and shifted to 24 for captive populations.
Avoidance of Inbreeding

During captivity, Maximum Avoidance of Inbreeding (MAI) regimes underwent a migration protocol in which randomized birth order in the population determined the zoo (captive deme) to which an individual would be shipped. For example, the third individual born in a zoo was transferred to the third of ten zoos. Random mating then occurred within demes. This created panmixia for maximal inbreeding avoidance.

Conditions on Release

After captivity, each regime was transferred to a single deme and allowed to outcross. The following generation, the individuals were seeded into 4 novel wild demes and allowed to breed until the population reached carrying capacity. Populations were then allowed to migrate to the remaining open wild demes during which m was set to 0.05. Phenotypic optimum in any deme was determined randomly between the values of 0 to 32 and allowed to shift in any generation by +/- 4. This created a temporal autocorrelation while simulating a variable environment.

Statistical analysis

The differences between paired replicates were analyzed using paired t-tests (Microsoft Excel). Inbreeding coefficient measures were calculated using Genetic Data Analysis (Lewis and Zaykin’s 2002).
CHAPTER III

RESULTS

Figure 3. Fitness shown for MAI and DS treatments during captivity. Paired T-test with Bonferroni correction was used to test for significant differences between treatments. Adjusted alpha value was .01. Error bars were calculated using the 95% C.I. ‘*’ shows statistical significance.
Figure 4. Average $F_{in}$ for DS and MAI populations during captivity. Paired T-test with Bonferroni correction was used to test for significant differences between treatments. Adjusted alpha value was .0125 and was calculated using both captive and release generations. Error bars were calculated using the 95% C.I. ‘*’ shows statistical significance. Non-significant values not shown.

Figure 3 shows relative fitness for captive populations that were reared under a phenotypic optimum shifted from the original wild peak (trait value = 16) by 50% of the maximum allele value (trait value = 24). Populations reared under this phenotypic shift allow for the possibility of adaptation to a captive environment. Until release, MAI populations had higher average fitness. After release the DS populations showed a marginally higher average fitness compared to MAI populations (p=0.067).

In both simulations inbreeding load followed the same pattern. DS populations maintained higher inbreeding loads compared to MAI populations. All populations are outcrossed during preparation for release and DS populations are able to get the release
from inbreeding that MAI populations experienced as soon as they were placed into the zoo (fig 4). During captivity and after the initial release from inbreeding load, MAI populations begin to build load due to the relatively small populations sizes (100 total individuals across all zoos).

Genetic variability was measured via the average heterozygosity and $F_{st}$ at both inbreeding and QTL loci. Because no difference existed between inbreeding and QTL measures, only the QTL $F_{st}$ values were shown (figure 5). MAI populations attained high (~45%) heterozygosity when outcrossed at the onset of captivity. DS populations attained their highest heterozygosity (~40%) when allowed to outcross in generation twenty. Subsequently, $F_{st}$ values for DS populations remained higher than MAI values until outcrossed (fig 5). $F_{st}$ values were near equal in the outcrossed generation before DS populations acquired and maintained significantly lower $F_{st}$ values for the duration of the simulation.

Fitness after release showed a marginally significant ($p=0.06206$) fitness advantage for DS populations (fig 6). This advantage was maintained until 60-70 generations after release and regained by the end of the simulation. This correlated to slight increases in population size and number of demes for DS populations (fig 8). Average heterozygosity remained higher after release for DS populations than for MAI populations and correlated to lower $F_{st}$ measures (fig 7). By twenty generations after release, DS populations maintained significantly lower $F_{st}$ measures.
Figure 5. Average $F_{st}$ measures for DS and MAI populations in captivity. Paired T-tests with Bonferroni correction were used to test for significant differences between treatments. Adjusted alpha value was .004. Error bars were calculated using the 95% C.I. *$p$* shows statistical significance. Non-significant values not shown.
Figure 6. Average fitness measures for DS and MAI populations after release. Differences between treatments were tested using a repeated measures ANOVA. Paired t tests showed no significant differences between treatments within a single generation. DS populations had significantly higher fitness ($p=.002442$). Error bars were calculated using the 95% C.I.
Figure 7. Average $F_{st}$ for DS and MAI populations after release. All differences in $F_{st}$ values are significant at ten generations post release. Error bars were calculated using the 95% C.I. ‘*’ shows statistical significance.
Figure 8. Census for DS and MAI populations after release. No significant differences between population sizes occur before 50 generations post release though DS populations maintain a graphical advantage throughout (paired t test). Error bars were calculated using the 95% C.I.
CHAPTER IV
DISCUSSION

*Success of released populations*

If release back into the wild is the ultimate goal of a breeding program, then both the long-term success and early proliferation of released populations are critical management goals. We found that upon release into novel and variable environments, populations coming from DS treatments tended to spread faster (figure 7) than populations from MAI treatments over the first 50 generations post release. In the most sensitive first 10 generations the DS treatments performed as well as MAI treatments with no significant difference between the breeding strategies. A difference between treatments that became evident by 30 generations post release allowed the DS populations to spread to a ~4% larger geographic range than the MAI populations with ~7% more individuals by 50 generations after release (fig 7). The critical period of any recovery program when reintroduction is initiated but restoration is far from complete has been understudied. We have shown that MAI treatments fair no better than DS treatments, and that the downstream effects of captive breeding in the simulations have a compounding effect on population viability.

Increased success after release is tied to several factors. By all measures when released into a variable environment, DS proved to be the superior breeding strategy in terms of individual fitness. Average individual fitness was higher for the first 30
generations in the DS treatment than in the MAI treatment. Several factors can influence the mean individual fitness of populations including adaptive optimization to current conditions, genetic variability to accommodate changing environments, and inbreeding depression. In our model it is expected that as populations colonize new demes that individuals will have lower fitness relative to individuals that remain in their parent deme. For this reason individual fitness may not reflect the population level success, as fitness was calculated based on local trait optima. Fitness tended to be higher for DS treatments until 40 generations post release and maintained that advantage for 20 more generations. The higher fitness of DS treatments immediately after release did correlate to higher populations and larger geographical area colonized.

After release DS treatments tended to have significantly higher heterozygosity and subsequently lower Fst values. This genetic variation is at the level of the loci and is precisely what MAI strategies aim to maximize during captivity (Meffert et al. 2005, Frankham 2006), but upon release both populations are released from all inbreeding and DS populations are left with greater variation leading to increased heterozygosities. Presumably this within individual variation as well as the variation between captive demes allows for an adaptive advantage when released, especially when the release environments are variable or unpredictable. This correlated with the higher fitness, population, and number of demes colonized for DS treatments. This is the strongest evidence demonstrating that the loss of variation via inbreeding in captivity can be a transient or apparent loss.
The other chief concern for captive breeding success is the impact of inbreeding depression on the fitness of released populations. Figure 6 shows that DS treatments maintained significantly lower $F_{st}$ values immediately after release. The difference between treatments was only marginally significant for the first ten generations post release. The release from inbreeding depression allotted to MAI treatments at the beginning of captivity was not maintained after captivity. However, that same release allotted to DS treatments at the end of captivity proved to be a significant factor in their higher fitness measures. Some component of the higher $F_{st}$ values for MAI treatments after release may be attributed to the slower geographical spread of MAI populations due presumably to lower genetic variation. Slower spreading populations will have higher $F_{st}$ values due to the more limited opportunities to outcross. The findings coincide with predictions for the SP strategies that were modified in this experiment (Templeton and Reed 1984, Crnokrak and Barrett 2002).

Success in captivity

In captivity, fitness and fecundity are the primary goals for species. We found that in captivity and as expected MAI populations showed signs of success. MAI populations maintained significantly higher heterozygosity, lower $F_{st}$ measures and maintained higher fitness. The higher average heterozygositites correlated with an increased response to captive selection in MAI treatments. This response was met with a correlated increase in fitness for each generation in captivity. $F_{is}$ measures demonstrated that inbreeding load was also lower in MAI populations, further contributing to the higher captive fitness. However, it is when all treatments experience outcrossing and reintroduction into the
wild where the many aforementioned advantages that MAI treatments exhibit are either lost or insignificant when compared to DS treatments. This further emphasizes the fact that advantages while in captivity often do not correlate to the release environment, and that adaptation to captive environments is a growing concern for captive breeders contrary to many of the findings sensu Meffert. Additionally, as many have demonstrated, adaptation to captive environments is rapid (Briscoe et al. 1992, Gilligan and Frankham 2003, Frankham 2008, and Christie et al. 2012) and subsequently maladaptive for released populations.

**Transition to the wild**

Outcrossing prior to release for DS treatments allowed for protection from adaptation to the captive environment and the subsequent preservation of alleles otherwise lost during MAI treatments. The increased homozygosity among DS treatments accounts for this finding as well as the lowered inbreeding load from exposure to selection of the homozygous deleterious recessive alleles. Fitness in captivity after outcrossings (MAI in generation one and DS in generation twenty in figure 2) yielded nearly identical values indicating that DS populations did not suffer from the delayed outcrossing. $F_{is}$ measures (figure 3) indicated that neither treatment suffered tremendously during captivity as levels were restored to pre-captive levels when outcrossed prior to release. Thus the impetus for MAI strategies sensu Meffert and Frankham is invalid and the only benefit for MAI strategies as made evident by this study is an increase in captive fecundity.
However heterozygosity never reached a higher value than when MAI treatments were crossed in the first generation of captivity and all demes were released from the population structure of the natal environment. The loss of this heterozygosity among QTLs in MAI populations continued throughout captivity and release. This as well as the degree to which MAI treatments adapted to captive environments further indicates that length of captivity is vitally important to consider with MAI treatments (Christie et al. 2012), where as with DS treatments there does not seem to be as much cause for concern due to the restoration of values when outcrossed. $F_{st}$ values (figure 4) decreased tremendously for MAI populations when outcrossed, but not as much as when DS populations were outcrossed prior to release. In this case captivity was beneficial for both treatments, potentially demonstrating any advantages that captive strategies seem to have while in captivity.
CHAPTER V
SUMMARY AND CONCLUSIONS

Within captive environments, treatments went as commonly predicted (Frankham 1995, 2006; Rudnick and Lacy 2008, Meffert et al. 2005). MAI strategies were superior in terms of fitness within the zoo, maintained a lower average inbreeding load, lower $F_{st}$ measures, and maintained higher heterozygosity among QTLs. These advantages over DS counterparts were lost when treatments were crossed before release. After outcrossing, DS treatments acquired an advantage in terms of fitness, inbreeding load, $F_{st}$ and heterozygosity. This resulted in faster population growth and faster colonization of novel wild demes.

DS populations show higher fitness relative to MAI populations after release despite the fact that MAI populations maintained an advantage in captivity. Fitness, inbreeding load, and $F_{st}$ do not extrapolate from the captive environment as may have been predicted without taking into account the population structure and the population’s underlying genetic variation. Heterozygosity and $F_{st}$ in captivity did correlate with fitness and inbreeding load. However, that was overcome by outcrossing before release. Levels of heterozygosity within treatments lowered as populations adapted to release environments and correlated with levels at release. Thus, heterozygosity was a type of genetic currency used to explore and colonize novel wild demes. If used to adapt to zoos, then that heterozygosity is no longer available when released into the wild.
One of the assumptions of many breeding regimes is that the environment of the zoo or captive environment mimics the wild natal environment. Thus they assume that adaptation to the captive environment is not as deleterious as the load attributed to inbreeding. An additional assumption tested by this simulation was the heterogeneity of the release environment. Given that species often cannot return from whence they came, placing captive species in variable environments tests whether the genetic variation as measured by heterozygosity, or $F_{st}$ is a better predictor of fitness and population growth after release. We demonstrated that $F_{st}$ associated with population structure is a superior measure when dealing with long-term species success. $F_{st}$ represents only the loss of heterozygosity that is due to population structure. That lost variation is the amount of variation preserved by local breeding that can be released later when the population is outcrossed.

The enhanced ability of the DS populations to conquer novel demes with presumably novel peaks, and the relative benefits in terms of fitness fit the models of Moore (1996) and reflect a mechanism that can drive variance induced peak shifts (Whitlock 1995). The higher variance within the DS populations (between zoos) potentially allowed for more peak shifts within the variable environment, increasing their adaptability and inevitably leading towards sustainability of the released species. Though Chesser et al. (1980) predicted that genetic variation could be maintained when migration between demes was pulsed at intermediate rates, and many have argued that variation is key for long-term survival (Frankham 1995, 2006; Rudnick and Lacy 2008, Meffert et al. 2005), few have looked at captive dynamics that take into account both of these ideas.
Goldstein and Holsinger's (1992) notion of genetic redundancy allows for a metapopulation level approach to captive breeding, which can be pragmatic considering that captive bred programs can vary the migration between zoos as well as other parameters that control inbreeding levels. Not only can one see that the variation captured between zoos as predicted by GR is functional, but one can see that the more structured DS populations preserved this variation best.

Optimizing species for the zoo environment is a trade-off between individual fitness and adaptability. MAI regimes hedge their bets on the fitness of the individual during captivity. DS regimes look for genetic variance after release, and provide for increases in fitness and lower inbreeding load in the novel environment. The advantages attributed to MAI strategies are only evident under a captive regime and have little bearing on long-term adaptability or fitness according to our simulation.

Different rates of outcrossing outside of the brief investigation above need to be investigated in greater detail in order to understand how the captive population structure impacts the genetics when released. Additionally, which species will are best suited for DS or MAI strategies has yet to be determined. The historical population structure will have tremendous impact on the levels of inbreeding and population decline that can occur in captive species. This was evident in the statistical analysis. However, the above results are generalizable given that inbreeding and outcrossing need not occur as absolutes, and should probably be tailored to the individual needs of the breeding program.
REFERENCES


APPENDIX A

SOURCE CODE FOR SEED POPULATIONS

The following program is the simulation written in FORTRAN that utilized the associated parameter file to create the seed populations for the experiment.

* PHASE I
* Title: Maintenance of Variation During Captive Breeding in a Finite Number of Demes

* Version 1.3.final
* This program is intended to simulate the change in gene frequency of multiple demes under selective pressure at a number of epistatic loci.
* Small portions of this program were borrowed from a program entitled "Gametophytic Incompatibility" which was written by Kieth Crandall (Jan. 5th, 1987). The function ran2 was acquired from B.S. Wier's Genetic Data Analysis, 1990. Sinauer Assoc. Ran2 was modified by F. Moore June 1991.

* Author: Drew Jones & Francisco Moore, June 18, 2012
* Version without inbred fubar ***Drew

* Declarations:

INCLUDE [DREW.FORTRAN.EXPERIMENT]16LOCUS1PEAKCALL.FOR'
* The following arrays are used to store genotypes by deme, individual, locus, and chromosome (Genotypic arrays for each stage).
INTEGER Migrant (maxdeme,maxheadroom,maxchromo,maxlocus)
INTEGER Adolescent (maxdeme,maxheadroom,maxchromo,maxlocus)
INTEGER Young (maxdeme,maxheadroom,maxchromo,maxlocus)
* The following arrays are used to store a sex by deme, and individual (sex arrays for each stage).
INTEGER SexY (maxdeme,maxheadroom)
INTEGER SexA (maxdeme,maxheadroom)
INTEGER SexM (maxdeme,maxheadroom)
* The following arrays are used to keep track of the number of individuals within a deme
INTEGER HeadcountY (maxdeme)
INTEGER HeadcountA (maxdeme)
INTEGER HeadcountM (maxdeme)
* The following variables will be used to keep track of cells of the different arrays, throughout the program.
INTEGER IdemeY
INTEGER IdemeM
INTEGER IdemeA
* Totind is a variable which counts the total number of individuals
Which have been produced in all demes over all generations
INTEGER totind
* The following variables control loops

INTEGER igen
INTEGER iskip
INTEGER genwrt
INTEGER ITRIAL
* ProbG is the probability of migrating a distance using a gamma distribution
REAL ProbG (0:maxXdist)
* The following arrays represent numbers of demes at a distance
INTEGER DX (0:xmax, maxnumb)
INTEGER DY (0:xmax, maxnumb)
INTEGER Numb (0:xmax)
INTEGER count

PARAMETER (iskip =10000)

* Program execution begins at this point
*
******************************************************************************
* IDUM seeds the random number generator and should be a large negative number.
* The value for IDUM could be read off the clock.
* secds is a VAX specific command that returns the clock reading in seconds which is here
* used as a seed value if this command is not available a large negative number should be
* used as a seed.
******************************************************************************
* IDUM = -1 * secds(0.00)
IDUM2 = IDUM
ISEED = IDUM
* Open Files Routine
* this is a routine which opens output files for storage of data
* and then writes the headers for the files

OPEN (unit=16,file=
+ '[DREW.FORTRAN.EXPERIMENT]popstats.dat',
+ status='new',access='sequential',form='formatted',recl=2048)

OPEN (unit=20,file=
+ '[DREW.FORTRAN.EXPERIMENT]popdeme1.dat',
+ status='new',access='sequential',form='formatted',recl=2048)

OPEN (unit=60,file=
+ '[DREW.FORTRAN.EXPERIMENT]poprand1.dat',
+ status='new',access='sequential',form='formatted',recl=2048)
OPEN (unit=21, file=
+ '[DREW.FORTRAN.EXPERIMENT]popdeme2.dat',
  status='new', access='sequential', form='formatted', recl=2048)

OPEN (unit=61, file=
+ '[DREW.FORTRAN.EXPERIMENT]poprand2.dat',
  status='new', access='sequential', form='formatted', recl=2048)

OPEN (unit=22, file=
+ '[DREW.FORTRAN.EXPERIMENT]popdeme3.dat',
  status='new', access='sequential', form='formatted', recl=2048)

OPEN (unit=62, file=
+ '[DREW.FORTRAN.EXPERIMENT]poprand3.dat',
  status='new', access='sequential', form='formatted', recl=2048)

OPEN (unit=23, file=
+ '[DREW.FORTRAN.EXPERIMENT]popdeme4.dat',
  status='new', access='sequential', form='formatted', recl=2048)

OPEN (unit=63, file=
+ '[DREW.FORTRAN.EXPERIMENT]poprand4.dat',
  status='new', access='sequential', form='formatted', recl=2048)

OPEN (unit=24, file=
+ '[DREW.FORTRAN.EXPERIMENT]popdeme5.dat',
  status='new', access='sequential', form='formatted', recl=2048)

OPEN (unit=64, file=
+ '[DREW.FORTRAN.EXPERIMENT]poprand5.dat',
  status='new', access='sequential', form='formatted', recl=2048)

OPEN (unit=25, file=
+ '[DREW.FORTRAN.EXPERIMENT]popdeme6.dat',
  status='new', access='sequential', form='formatted', recl=2048)

OPEN (unit=65, file=
+ '[DREW.FORTRAN.EXPERIMENT]poprand6.dat',
  status='new', access='sequential', form='formatted', recl=2048)

OPEN (unit=26, file=
+ '[DREW.FORTRAN.EXPERIMENT]popdeme7.dat',
  status='new', access='sequential', form='formatted', recl=2048)

OPEN (unit=66, file=
+ '[DREW.FORTRAN.EXPERIMENT]poprand7.dat',
  status='new', access='sequential', form='formatted', recl=2048)

OPEN (unit=27, file=
+ '[DREW.FORTRAN.EXPERIMENT]popdeme8.dat',
  status='new', access='sequential', form='formatted', recl=2048)

OPEN (unit=67, file=
+ '[DREW.FORTRAN.EXPERIMENT]poprand8.dat',
  status='new', access='sequential', form='formatted', recl=2048)
OPEN (unit=28,file= + '[DREW.FORTRAN.EXPERIMENT]popdeme9.dat', + status='new',access='sequential',form='formatted',recl=2048)

OPEN (unit=68,file= + '[DREW.FORTRAN.EXPERIMENT]poprand9.dat', + status='new',access='sequential',form='formatted',recl=2048)

OPEN (unit=29,file= + '[DREW.FORTRAN.EXPERIMENT]popdeme10.dat', + status='new',access='sequential',form='formatted',recl=2048)

OPEN (unit=69,file= + '[DREW.FORTRAN.EXPERIMENT]poprand10.dat', + status='new',access='sequential',form='formatted',recl=2048)

WRITE (16,1005)(1-P),maxqlocus,maxinbred,k,s,l, + idum,idum2,iseed

WRITE (16,1007)

1004 FORMAT(3x,'z=0',4x,'z=1',4x,'z=2',4x,'z=3',4x,'z=4',4x,'z=5', + 4x,'z=6',4x,'z=7',4x,'z=8',4x,'z=9',4x,'z=10',4x,'pwreal',3x, + '# indiv',5x,'gen',3x,'itrial')

1006 FORMAT(3x,'z=0',4x,'z=1',4x,'z=2',4x,'z=3',4x,'z=4',4x,'z=5', + 4x,'z=6',4x,'z=7',4x,'z=8',4x,'z=9',4x,'z=10',4x,'dwreal',4x, + 'deme',3x,'itrial')

1005 FORMAT('2-1-10 ','migration=',f10.8,1x,'#Quantitative loci=', + i3,'#inbreeding loci=',i3,1x,'k=',f6.4,1x,'l=',f6.4,1x, + 's=',f6.4,1x,'IDUM=',i8,1x,'IDUM2=',i8,1x,'ISEED=',i8, + 'Quantitative peak=16*)

1007 FORMAT(5x,'trial',1x,'Gen',3x,'Deme',2x,'Indiv',2x,'Fitness', + 11x,'Het',3x,'DR',1x,'QT1',3x,'QT2',3x,'QT3',3x,'QT4',3x, + 'QT5',3x,'QT6',3x,'QT7',3x,'QT8',3x,'QT9',3x,'QT10',2x,'QT11' + 2x,'QT12',2x,'QT13',2x,'QT14',2x,'QT15',2x,'QT16',2x,'DEL...')

* ******************************************************
* Here is a good spot to define the probabilities of Gamma Function probg
* ******************************************************

CALL GAMMA MIGRATION (probg)

* ******************************************************
* Now the number of demes which are located a given distance away from a given
deme need to determined for use during periods of migration.
* ******************************************************
CALL NUMBDIST (Numb, DX, DY)

* loop through #? trials
* DO 886 itrial=1,1
  igen=0

* Now it is time too seed the parental Genotypes, and initialize the adolescent Genotypes
* CALL PARSEED (Sexy,Young,Sexa,Adolescent,HeadcountA,HeadcountY, totind,IDUM,IDUM2,ISEED)

* Now, the statistics on the parental generation can be done (see flow chart C)
* CALL STAT (igen, young,totind,headcountY,iskip,itrial)

* Now it is time to begin looping through the generations
* DO 100 igen = 1 , maxgen

* The youngsters now go through a bout of selection.
* CALL SELECTION (Adolescent, Young, HeadcountY, HeadcountA, SexY, SexA, IDUM,IDUM2,ISEED)

* It is time to Initialize (re-initialize) the migrant demes so they will be ready for new migrants.
* CALL INITIALIZE MIGRANT (Migrant, HeadcountM, SexM)

* Now that we have initialized the migrant array migration can take place
* CALL MIGRATION (SexM, SexA, Adolescent, Migrant, HeadcountM, IDUM,IDUM2,ISEED, Numb, DX, DY, Probg, HeadcountA)

* Now the young and Adolescent Arrays need to be reset
* CALL INITIALIZE Y AND A (Young, Adolescent, HeadcountY, HeadcountA, SexY, SexA)

* Mating time
* CALL MATING (Migrant, Young, SexM, SexY, totind, HeadcountM, HeadcountY, IDUM,IDUM2,ISEED)
CALL STAT (igen, young, totind, headcountY, iskip, itrial)
CALL STAT2 (igen, young, totind, headcountY, iskip, itrial)

IF(igen .eq. maxgen)THEN
DO 111 count=1, 10
  type*, count
    CALL ZOOSEED (Sexy, Young, Sexa, Adolescent, HeadcountA,
                  + HeadcountY, totind, IDUM, IDUM2, ISEED, count)
    CLOSE(count+19)
    CLOSE(count+59)
  111 CONTINUE
END IF

CLOSE(itrial+20-1)
CLOSE(itrial+60-1) artifacts from a longer running program damnit!!

886 CONTINUE
END

SUBROUTINE PARSEED (Sexy, Young, Sexa, Adolescent, HeadcountA,
   + HeadcountY, totind, IDUM, IDUM2, ISEED)

INCLUDE + '[DREW.FORTRAN.EXPERIMENT]16LOCUS1PEAKCALL.FOR'

The following variables and parameters are the same as in the main program

INTEGER Adolescent (maxdeme, maxheadroom, maxchromo, maxlocus)
INTEGER YounG(maxdeme, maxheadroom, maxchromo, maxlocus)
INTEGER SEXY (maxdeme, maxheadroom)
INTEGER SEXA (maxdeme, maxheadroom)
INTEGER Headcounty (maxdeme)
INTEGER Headcounta (maxdeme)
INTEGER Idemey
INTEGER Indivy
INTEGER Icopyy
INTEGER locusy
INTEGER Idemea
INTEGER Icopya
INTEGER Indiva
INTEGER Locusa
INTEGER totind
INTEGER IOREO

This Subroutine seeds the genotypes and sexes of the parental generation
see flow chart B
The first thing to do is set the population counters to zero
\[ \text{totind} = 0 \]

Now loop through and create the proper number of individuals in each population and flag all non-existant individuals with -1 for young and adolescents

```plaintext
DO 20 idemeY = 1, maxdeme !loop through all demes
   idemea = idemeY
   HeadcountY (idemeY) = 0
   headcountA (idemeA) = 0
   DO 30 IndivY = 1, maxheadroom !loop through all individuals
      indivA = indivY
      SexA (idemeA, indivA) = -1 ! adolescents don't exist yet
      IF (indivY.le.(maxindiv)) THEN
         SexY (idemeY, indivY) = (IRBIT2(ISEED) + 1)
         DO 40 icopyY = 1, maxchromo ! we need to seed values at each copy
            icopyA = icopyY
         DO 50 locusY = 1, maxlocus ! and each locus
            locusA = locusY
            IOREO = RAN1(IDUM) * MUTATIONRATE + 1
            IF (locusY .le. (maxqlocus)) THEN
               IF (IOREO .ge. (mutationrate)) THEN
                  Young (idemeY, indivY, icopyY, locuscY) = 1
               ELSE
                  Young (idemeY, indivY, icopyY, locuscY) = 2
               END IF
            ELSE
               Young (idemeY, indivY, icopyY, locuscY) = 2
            END IF
         END IF
      ELSE
      END IF
      Adolescent (idemeA, indivA, icopyA, locuscA) = -1 ! adolescents don't exist yet
   CONTINUE
30 CONTINUE
20 CONTINUE
```

```plaintext
HeadcountY (idemeY) = HeadcountY (idemeY) + 1 ! a new individual was just created
totind = totind + 1
ELSE ! these youngsters don't exist yet
   SexY (idemeY, indivY) = -1 ! -1 indicates that the individual is nonexistent
   DO 60 icopyY = 1, maxchromo
      icopyA = icopyY
   DO 70 locusY = 1, maxlocus
      locuscA = locuscY
      Young (idemeY, indivY, icopyY, locuscY) = -1 ! these youngsters don't exist yet
   Adolescent (idemeA, indivA, icopyA, locuscA) = -1 ! adolescents don't exist yet
   CONTINUE
70 CONTINUE
60 CONTINUE
30 CONTINUE
20 CONTINUE
```
SUBROUTINE ZOOSEED (Sexy, Young, Sexa, Adolescent, HeadcountA, HeadcountY, totind, IDUM, IDUM2, ISEED, count)

INCLUDE + ['[DREW.FORTRAN.EXPERIMENT]16LOCUS1PEAKCALL.FOR'

* ***************************************************************
* The following variables and parameters are the same as in the main program
* ***************************************************************

INTEGER Adolescent (maxdeme, maxheadroom, maxchromo, maxlocus)
INTEGER Young(maxdeme, maxheadroom, maxchromo, maxlocus)
INTEGER Young2(maxdeme, maxheadroom, maxchromo, maxlocus)
INTEGER SEXY (maxdeme, maxheadroom)
INTEGER zoogen1(0:(2*maxlocus-1))
INTEGER zoogen2(0:(2*maxlocus-1))
INTEGER ib
INTEGER ic
INTEGER j1
INTEGER j
INTEGER j2
INTEGER SEXA (maxdeme, maxheadroom)
INTEGER Headcounty (maxdeme)
INTEGER Headcounta (maxdeme)
INTEGER Idemey
INTEGER Indivy
INTEGER Icopyy
INTEGER Locusy
INTEGER Idemea
INTEGER Icopya
INTEGER Indiva
INTEGER Locusa
INTEGER totind
INTEGER IOREO2
INTEGER IOREO3
INTEGER IOREO4
INTEGER IOREO5
INTEGER unitd
INTEGER unitr
INTEGER count

unitd = (count+19)
unitr = (count+59)

DO 4101 idemey=1,maxzoodeme
IOREO2 = RAN1(IDUM) * (maxdeme) + 1
DO 4102 indivy=1,maxzooheadroom
DO 9251 j=0,(2*maxlocus)
zoogen1(j)=0
9251 CONTINUE
IOREO3 = RAN1(IDUM) * headcounty(IOREO2)+1

ib=-1
   DO 4103 locusY=1,maxlocus
       DO 4104 icopyY=1,maxchromo
           ib=ib+1
           zoogen1(ib)=Young(ioreo2,ioreo3,icopyY,locusY)
           IF(zoogen1(ib).eq.-1)THEN
               IOREO3 = RAN1(IDUM) * headcounty(IOREO2)+1
               zoogen1(ib)=Young(ioreo2,ioreo3,icopyY,locusY)
           END IF
           CONTINUE
  4104
   CONTINUE
  4103
   CONTINUE
   WRITE(unitd,9000)idemey,indivY,sexY(IOREO2,IOREO3),+
       (zoogen1(j1),j1=0,231)
  4102
   CONTINUE
  4101

   DO 4106 idemey=1,maxzoodeme
       DO 4107 indivy=1,maxzooheadroom
           DO 9252 j=0,(2*maxlocus)
               zoogen2(j)=0
           CONTINUE
           9252
   CONTINUE
   IOREO4 = RAN1(IDUM) * (maxdeme)+1
   IOREO5 = RAN1(IDUM) * headcounty(IOREO4)+1
   ic=-1
   DO 4108 locusa=1,maxlocus
       DO 4109 icopya=1,maxchromo
           ic=ic+1
           zoogen2(ic)=Young(ioreo4,ioreo5,icopya,locusY)
           IF(zoogen2(ic).eq.-1)THEN
               IOREO5 = RAN1(IDUM) * headcounty(IOREO4)+1
               zoogen2(ic)=Young(ioreo4,ioreo5,icopyY,locusY)
           END IF
           CONTINUE
  4109
       CONTINUE
  4108
   CONTINUE
   WRITE(unitr,9000)idemey,indivy,sexY(IOREO4,IOREO5),+
       (zoogen2(j2),j2=0,231)
  4107
   CONTINUE
  4106
   CONTINUE
  9000 FORMAT(I5,',',I5,',',I2,','265(1x,I2))
   END

SUBROUTINE NUMBDIST(Numb, DX, DY)
Now the number of demes which are located a given distance away from a given deme need to be determined for use during periods of migration. This subprogram stores the number \((Numb(ir))\) of demes which are a given distance \((ir)\) away from any deme. It also stores the coordinates \((DX(ir,Numb(ir)),DY(ir,Numb(ir)))\) of the different demes relative to the originating deme \((\text{Deme of origin} = (0,0))\).

The following variables represent distances and what not which are used in calculating the number of demes at a distance:

- \(\text{INTEGER } ir\)
- \(\text{INTEGER } a\)
- \(\text{INTEGER } b\)
- \(\text{REAL*8 } c\)
- \(\text{REAL*8 } d\)
- \(\text{INTEGER } i\)
- \(\text{INTEGER } j\)

\[\begin{align*}
\text{DO } 100 & \text{ I=0,XMAX} \\
\text{DO } 200 & \text{ J=1, MAXNUMB} \\
\text{DX}(i, J) & = 0 \\
\text{DY}(i, J) & = 0 \\
\text{200 } & \text{ CONTINUE} \\
\text{100 } & \text{ CONTINUE}
\end{align*}\]

\[\begin{align*}
\text{IF } (\text{MOD(xmax,2).NE.0}) & \text{ THEN } \text{IF xmax is odd -maxxdist to maxxdist won't overlap} \\
\text{IF } (\text{MOD(ymax,2).NE.0}) & \text{ THEN } \text{IF ymax is odd -maxydist to maxydist won't overlap} \\
\text{DO } 120 & \text{ i = -maxXdist, maxXdist} \quad \text{! loop through all demes} \\
\text{DO } 130 & \text{ j = -maxYdist, maxYdist} \quad \text{! in all directions} \\
a & = i \quad \text{! calculate the integer distance to this deme} \\
b & = j \quad \text{!} \\
c & = ((a*a) + (b*b)) \quad \text{!} \\
d & = \sqrt{c} \quad \text{!} \\
ir & = \text{INT}(d) \quad \text{! calculate the integer distance to this deme} \\
\text{IF } (\text{ir .gt. maxXdist}) & \text{ IR = maxXdist} \quad \text{! anything greater than a half of the way around the} \\
\text{torus is equally far} \\
\text{Numb(ir) = Numb(ir) + 1} \quad \text{! One more at this distance} \\
\text{DX(ir, Numb(ir)) = i} \quad \text{!assign these coordinates to this number deme} \\
\text{DY(ir, Numb(ir)) = j} \\
\text{130 } & \text{ CONTINUE} \\
\text{120 } & \text{ CONTINUE} \\
\text{ELSE } & \text{! if Ymax is even you don't want to overlap map edges} \\
\text{DO } 140 & \text{ i = -maxXdist, maxXdist} \quad \text{! loop through all demes} \\
\text{DO } 150 & \text{ j = -maxYdist, maxYdist-1} \quad \text{! in all directions} \\
a & = i \quad \text{! calculate the integer distance to this deme} \\
b & = j \quad \text{!} \\
c & = ((a*a) + (b*b)) \quad \text{!}\end{align*}\]
d = SQRT (c)  ! calculate the integer distance to this deme
ir = INT (d)  ! anything greater than a half of the way around the torus is equally far
* NuLl(ir) = NuLl(ir) + 1  ! One more at this distance
DX(ir, NuLl(ir)) = i  ! assign these coordinates to this number deme
DY(ir, NuLl(ir)) = j
150 CONTINUE
140 CONTINUE
END IF
ELSE  ! if Xmas is even you don't want to overlap map edges
IF (MOD(ymax,2) .NE. 0) THEN  ! if ymax is odd -maxydist to maxydist won't overlap
DO 160 i = -maxXdist, maxXdist-1  ! loop through all demes
  DO 170 j = -maxYdist, maxYdist  ! in all directions
    a = i  ! calculate the integer distance to this deme
    b = j  !
    c = ((a*a) + (b*b))  !
    d = SQRT (c)  !
    ir = INT (d)  ! calculate the integer distance to this deme
    IF (ir .gt. maxXdist) ir = maxXdist  ! anything greater than a half of the way around the torus is equally far
    NuLl(ir) = NuLl(ir) + 1  ! One more at this distance
    DX(ir, NuLl(ir)) = i  ! assign these coordinates to this number deme
    DY(ir, NuLl(ir)) = j
170 CONTINUE
160 CONTINUE
END IF
ELSE  ! if Ymax is also even you don't want to overlap map edges
DO 180 i = -maxXdist, maxXdist-1  ! loop through all demes
  DO 190 j = -maxYdist, maxYdist-1  ! in all directions
    a = i  ! calculate the integer distance to this deme
    b = j  !
    c = ((a*a) + (b*b))  !
    d = SQRT (c)  !
    ir = INT (d)  ! calculate the integer distance to this deme
    IF (ir .gt. maxXdist) ir = maxXdist  ! anything greater than a half of the way around the torus is equally far
    NuLl(ir) = NuLl(ir) + 1  ! One more at this distance
    DX(ir, NuLl(ir)) = i  ! assign these coordinates to this number deme
    DY(ir, NuLl(ir)) = j
190 CONTINUE
180 CONTINUE
END IF
END

SUBROUTINE SELECTION (Adolescent, Young, HeadcountY, HeadcountA, + SexY, SexA, IDUM, IDUM2, ISEED)

INCLUDE
+ '[DREW.FORTRAN.EXPERIMENT]16LOCUS1PEAKCALL.FOR'
INTEGER Adolescent (maxdeme,maxheadroom,maxchromo,maxlocus)
INTEGER Young (maxdeme,maxheadroom,maxchromo,maxlocus)
INTEGER SEXY (maxdeme, maxheadroom)
INTEGER SEXA (maxdeme, maxheadroom)
INTEGER Headcounty (maxdeme)
INTEGER Headcounta (maxdeme)
INTEGER Idemey
INTEGER Indivy

******************************************************************************
* These two variables are used to compare the probability of survival and
* chance
******************************************************************************
REAL prob
REAL killer

******************************************************************************
* In this subroutine individuals in the young array are exposed to genotype dependent
* selection. The selection probabilities are as follows: If a genotype has at least one
* dominant allele (2) at each locus then The probability of survival is (1 + K*S) * L , L
* is the survival probability of a doubly homozygous recessive, and all individuals with
* at least one dominant allele at only one of the two loci experiences a prob of (1-S)*L
* Those that live through selection will be placed in the same deme in the adolescent array
******************************************************************************
DO 220 IdemeY = 1, maxdeme !loop through all demes
DO 230 IndivY = 1, HeadcountY (idemeY) !loop through all individuals
CALL SURVIVAL PROBABILITY (prob, Young, idemey, indivy)
killer = RAN2(IDUM2)
If (killer .le. prob) CALL ADOLESCENT STORAGE (Young,
+ Adolescent, SexY, SexA, idemey, indivy, headcountA)
230 CONTINUE
220 CONTINUE

SUBROUTINE SURVIVAL PROBABILITY (prob, Young, idemey, indivy)

INCLUDE
+ '[DREW.FORTRAN.EXPERIMENT]16LOCUS1PEAKCALL.FOR'
******************************************************************************
* This subroutine calculates the genotype dependent survival probability of individuals,
* and is called by the selection subroutine.
******************************************************************************
INTEGER Iplace
REAL IP2
REAL Prob
Real Load

******************************************************************************
* The following variables and parameters are the same as in the main program
******************************************************************************
INTEGER Young (maxdeme,maxheadroom,maxchromo,maxlocus)
INTEGER Idemey
INTEGER Indivy
INTEGER Icopyy
INTEGER locusy
prob= 0
Iplace = 0
DO 320 locusY = 1, maxqlocus !find the genotype
DO 330 icopyY = 1, maxchromo
   Iplace=Iplace+Young(idemeY, indivY, icopyY, locusY)-1
   CONTINUE
330 CONTINUE
320 CONTINUE
   IP2=Iplace
   IF (Ip2 .ge. peak)THEN
      prob= ((1.0-(((ip2-peak)/peak)*S))*L)
   else
      prob= ((1.0-(((peak-ip2)/peak)*S))*L)
   endif
   Call INBREEDING LOAD (IP2, Young, Idemey, Indivy, Load)
   Prob=Prob-Load
END

Subroutine INBREEDING LOAD (IP2, Young, Idemey, Indivy, Load)

INCLUDE + '[DREW.FORTRAN.EXPERIMENT]16LOCUS1PEAKCALL.FOR'
*
* ************************************************************
* This subroutine calculates the individuals inbreeding load across all heterosis and deleterious
* recessive loci.
* It is called by the Survival Prob subroutine.
* ************************************************************

INTEGER IP2
REAL Load
INTEGER locount
INTEGER hetadv
INTEGER delrec
INTEGER Idemey
INTEGER hicount
INTEGER Indivy
INTEGER Hetstart
INTEGER Delrecstart
INTEGER Young (maxdeme,maxindiv,maxchromo,maxlocus)
   INTEGER sumtotal
   hetadv = 0
   delrec = 0
load = 1.01
Hetstart= (maxqlocus+1)
Delrecstart= (maxhetadv +1)
DO 301 locount = Hetstart, maxhetadv !counts the impact of all het adv loci
   IF (YOUNG(idemeY,indivY,1,locount) .eq. YOUNG(idemeY,indivY,2,locount)) THEN
      hetadv = hetadv + 1
   !type *, 'da fuck 1 ',hetadv
   ELSE
      ENDIF
301 CONTINUE

DO 302 hicount = Delrecstart,maxlocus
   IF ((YOUNG(idemeY, indivY, 1, hicount) .eq. 1) .and. (YOUNG(idemeY,indivY,2,hicount).eq. 1)) THEN
      delrec = delrec + 1
      !type *, 'hahahahahaha ',delrec
   END IF
CONTINUE

subtotal = (delrec + hetadv)
Load = (inbeffect * subtotal)
!type *, this is a long line to determine if this shit works ', load
   !type *, the effect is ', inbeffect
END

SUBROUTINE ADOLESCENT STORAGE (Young, Adolescent, SexY, SexA, +
   idemeY, indivY, headcountA)

   *-----------------------------------------------------------------------
   * this subroutine stores the young which has most recently made it through a bout of selection
   * into the adolescent array
   *-----------------------------------------------------------------------

   INCLUDE
+ 'DREW.FORTRAN.EXPERIMENT]16LOCUS1PEAKCALL.FOR'

   INTEGER Adolescent (maxdeme,maxheadroom,maxchromo,maxlocus)
   INTEGER Young (maxdeme,maxheadroom,maxchromo,maxlocus)
   INTEGER SEXY (maxdeme, maxheadroom)
   INTEGER SEXA (maxdeme, maxheadroom)
   INTEGER Headcounta (maxdeme)
   INTEGER Idemey
   INTEGER Indivy
   INTEGER Icopyy
   INTEGER locusy
   INTEGER Icopya
   INTEGER Indiva
   INTEGER Locusa
   idemeA = idemeY ! the deme of the individual does not change
   headcountA(idemea) = headcountA(idemea) + 1
   indivA = headcountA(idemea)
   SexA (idemeA, indivA) = SexY (idemey, indivy)
   DO 420 locusY = 1, maxlocus ! loop through each locus
      locusA = locusY
      DO 430 icopyY = 1, maxchromo ! and each chromosome
         icopyA = icopyY
         Adolescent (idemeA, indivA, icopyA, locusA) = Young ! and copy the values
            (idemeY, indivY, icopyY, locusY)
   420 CONTINUE
   430 CONTINUE
END

SUBROUTINE INITIALIZE MIGRANT (Migrant, HeadcountM, SexM)

   *-----------------------------------------------------------------------
   * this subroutine initializes the migrant array with null values so that migration between
   * adolescent demes may begin
   *-----------------------------------------------------------------------

   INCLUDE
INTEGER Migrant (maxdeme,maxheadroom,maxchromo,maxlocus)
INTEGER SEXM (maxdeme, maxheadroom)
INTEGER Headcountm (maxdeme)
INTEGER Idemem
INTEGER Indivm
INTEGER Icopym
INTEGER Locusm

DO 520 idemeM = 1, maxdeme    ! within each deme
    HeadcountM (idemeM) = 0
DO 530 indivM = 1, maxheadroom  ! for each indiv
    Sexm (idemeM, indivM) = -1    ! initialize the sex
DO 540 icopyM = 1, maxchromo    ! and for each copy
    DO 550 locusM = 1, maxlocus   ! and at each locus
        Migrant (idemeM, indivM, icopyM, locusM) = -1    ! initialize the genotype
      CONTINUE
    CONTINUE
  CONTINUE
  CONTINUE
END

SUBROUTINE MIGRATION (SexM, SexA, Adolescent, Migrant, headcountM,
+    IDUM,IDUM2,ISEED, Numb, DX, DY, ProbG, HeadcountA)

* ******************************************
*    this subprogram controls the random migration of adolescent individuals between arrays
*    to produce the migrant array of individuals.
* ******************************************

INCLUDE + '[DREW.FORTRAN.EXPERIMENT]16LOCUS1PEAKCALL.FOR'

INTEGER Adolescent (maxdeme,maxheadroom,maxchromo,maxlocus)
INTEGER SEXA (maxdeme, maxheadroom)
INTEGER Headcounta (maxdeme)
INTEGER Headcountm (maxdeme)
INTEGER Idemem
INTEGER Idemea
INTEGER Indiva

INTEGER IKEEP
INTEGER IKEEPD
INTEGER IDIST

INTEGER Dx (0:xmax, maxnumb)
INTEGER Dy (0:xmax, maxnumb)
INTEGER numb (0:xmax)
REAL ProbG (0:maxXdist)

REAL RDIST

indiva = (RAN2(IDUM2) * maxheadroom) + 1     !start with a random individual
DO 620 Ikeep = 1, maxheadroom     !loop through all possible individuals
    idemeA = (RAN2(IDUM2) * maxdeme) + 1    !start with a random deme
    DO 630 Ikeepd = 1, maxdeme    !loop through all the possible demes
        IF (Adolescent(idemeA, indivA, 1 ,1).NE. -1) THEN
            CALL DEME CHOICE (idemeA,idemeM, Numb, DX, DY,)
            +    IDUM,IDUM2,ISEED,ProbG,idist,rdist)   !Choose a deme to migrate to
            IF ((HeadcountM (idemeM)) + 1 .le. maxheadroom)   !make sure there is room
      CONTINUE
    CONTINUE
  CONTINUE
  CONTINUE
  CONTINUE
END

SUBROUTINE DEME CHOICE (idemeA, idemeM, Numb, DX, DY,)
+    IDUM,IDUM2,ISEED,ProbG,idist,rdist)
CALL MIGRANT STORAGE (Migrant, Adolescent, SexM, rdist, ! if there is room
* store migrant
+ SexA, headcountM, headcountA, idemeA, indivA, idemeM, idist)
ELSE
END IF
idemeA = idemeA + 1
IF (idemeA .gt. maxdeme) idemeA = 1
CONTINUE

630
idivA = indivA + 1
IF (idivA .GT. maxheadroom) indivA = 1
620
CONTINUE
END

SUBROUTINE DEME CHOICE (idemeA, idemeM, Numb, Dx, DY, IDUM,
+ IDUM2,ISEED,ProbG,idist, rdist)
*
*************************
***********************
*
CHOOSE A DEME AT A DISTANCE
*
*************************
***************
* INCLUDE
+ 'DREW.FORTRAN.EXPERIMENT]16LOCUS1PEAKCALL.FOR'
 INTEGER Idemem
 INTEGER Idemea
 INTEGER IX
 INTEGER IY
 INTEGER Y0
 INTEGER X0
 INTEGER I
 INTEGER NRAND
 INTEGER IDIST
 INTEGER Dx (0:xmax, maxnumb)
 INTEGER Dy (0:xmax, maxnumb)
 INTEGER numb (0:xmax)
 REAL RDIST
 REAL ProbG (0:maxXdist)
*
***********************************************************************
* THIS CODE DETERMINES THE X AND Y COORDINATES OF THE DEME FROM ITS DEME
* NUMBER
*
***********************************************************************
IF (MOD (idemeA, Xmax) .eq. 0) THEN
  Y0 = (idemea/xmax)
ELSE
  Y0 = (idemeA/Xmax) + 1
END IF
X0 = idemeA - ((Y0 -1) * xmax)

***********************************************************************
* THIS CODE PICKS A RANDOM DISTANCE BASED ON THE PROB. FUNCTION
* GENERATED BY THE GAMMA FUNCTION ROUTINE
***********************************************************************
Rdist = RAN1(IDUM)

DO 720 I = 0, maxXdist
IF (Rdist .le. ProbG(I)) THEN
  Idist = I
GOTO 730

END IF

720 CONTINUE

730 Nrand = (RAN2(IDUM2) * Numb(idist)) + 1 ! CHOOSE DEME AT THE GIVEN DISTANCE

*============================================================================
* LOCATE THE DEME CHOSEN
*============================================================================

iX = X0 + DX(idist, nrand)
iY = Y0 + DY(idist, nrand)
IF(iX .gt. Xmax) iX = iX - Xmax
IF(iX .lt. 1) iX = iX + Xmax
IF(iY .gt. Ymax) iY = iY - Ymax
IF(iY .lt. 1) iY = iY + Ymax
IdemeM = iX + (Xmax * (iY - 1))

END

SUBROUTINE MIGRANT STORAGE (Migrant, Adolescent, SexM, rdist, + SexA, headcountM, headcountA, idemeA, indivA, idemeM, idist)

*----------------------------------------------------------------------
* STORES THE INDIV AT THE CHOSEN DEME
*----------------------------------------------------------------------

INCLUDE +'[DREW.FORTRAN.EXPERIMENT]16LOCUS1PEACKALL.FOR'
INTEGER Migrant (maxdeme,maxheadroom,maxchromo,maxlocus)
INTEGER Adolescent (maxdeme,maxheadroom,maxchromo,maxlocus)
INTEGER SEXA (maxdeme, maxheadroom)
INTEGER SEXM (maxdeme, maxheadroom)
INTEGER Headcounta (maxdeme)
INTEGER HeadcountM (maxdeme)
INTEGER Idemem
INTEGER Indivm
INTEGER Icopym
INTEGER Locum
INTEGER Idemea
INTEGER Icopya
INTEGER Indiva
INTEGER Locua
INTEGER IDIST
REAL RDIST

HeadcountM(idemeM) = HeadcountM(idemeM) + 1
IndivM = headcountM(idemeM)
SexM (idemeM, indivM) = SexA (idemeA, indivA)
DO 820 locusA = 1, maxlocus
   locusM = locusA
   DO 830 icopyA = 1, maxchromo
      icopyM = icopyA
      Migrant (idemeM, indivM, icopyM, locusM) = + Adolescent (idemeA, indivA, icopyA, locusA)
   CONTINUE
830 CONTINUE
CONTINUE

END

SUBROUTINE GAMMA MIGRATION (ProbG)

INCLUDE + '[DREW.FORTRAN.EXPERIMENT]16LOCUS1PEAKCALL.FOR'
REAL ProbG (0:maxXdist)
REAL Q
INTEGER M
Do 920 M = 0, maxXdist
   Q = 1 - P
   IF (M .eq. 0) then
       ProbG(M) = Q**M * P
   ELSE
       ProbG(M) = Q**M * P + ProbG(M-1)
   END IF
920 CONTINUE
END

SUBROUTINE INITIALIZE Y AND A (Young, Adolescent, HeadcountY, HeadcountA, SexY, SexA)

INCLUDE + '[DREW.FORTRAN.EXPERIMENT]16LOCUS1PEAKCALL.FOR'
* *****************************************************************
* The following variables and parameters are the same as in the main program
* *****************************************************************
INTEGER Adolescent (maxdeme,maxheadroom,maxchromo,maxlocus)
INTEGER Young (maxdeme,maxheadroom,maxchromo,maxlocus)
INTEGER SEXY (maxdeme, maxheadroom)
INTEGER sexa (maxdeme, maxheadroom)
INTEGER Headcounty (maxdeme)
INTEGER Headcounta (maxdeme)
INTEGER Idemey
INTEGER Indivy
INTEGER Icopyy
INTEGER locusy
INTEGER Idemea
INTEGER Indiva
INTEGER Locusa

* *****************************************************************
* This Subroutine initializes the genotypes and sexes of the young and adolescent arrays see flow chart G
* *****************************************************************
* The first thing to do is set the population counters to zero

DO 5 Idemey = 1, maxdeme
   idemea = idemey
SUBROUTINE MATING (Migrant, Young, SexM, SexY, totind, + HeadcountM, HeadcountY, IDUM,IDUM2,ISEED)

INCLUDE + '[DREW.FORTRAN.EXPERIMENT]16LOCUS1PEAKCALL.FOR'
INTEGER totind
INTEGER IdemeM
INTEGER IdemeY
INTEGER IndIVY
INTEGER maxoffspring
INTEGER newind
INTEGER numbmales
INTEGER numbfemales
INTEGER male (maxheadroom, maxchromo, maxlocus)
INTEGER female (maxheadroom, maxchromo, maxlocus)
INTEGER SEXY (maxdeme, maxheadroom)
INTEGER SEXM (maxdeme, maxheadroom)
INTEGER HeadcountY (maxdeme)
INTEGER HeadcountM (maxdeme)
INTEGER Migrant (maxdeme,maxheadroom,maxchromo,maxlocus)
INTEGER Young (maxdeme,maxheadroom,maxchromo,maxlocus)
DO 105 idemeM = 1, maxdeme
     idemeY = idemeM
     CALL SEX SORTING (numbmales, numbfemales, Male, Female, + SexM, idemem, HeadcountM, Migrant)
     IF (numbmales .ne. 0) THEN
       IF (numbmales .ne. 0) THEN
         CALL PARENTAL PLANNING (HeadcountM, + Maxoffspring, idemem)
       ELSE
         maxoffspring = 0
       END IF
     ELSE
       maxoffspring = 0
     END IF
     DO 115 newind = 1, maxoffspring
       CALL MATCHMAKER(idemeM, IDUM,IDUM2,ISEED, + newind, numbfemales,numbmales, Male,
Female, Young, sexY, indivY)

totind = totind + 1
HeadcountY(idemY) = HeadcountY(idemY) + 1

CONTINUE

SUBROUTINE PARENTAL PLANNING (HeadcountM, Maxoffspring, idemem)

INCLUDE + [DREW.FORTRAN.EXPERIMENT]16LOCUS1PEAKCALL.FOR'
INTEGER HeadcountM (maxdeme)
INTEGER maxoffspring
INTEGER idemem
Maxoffspring = headcountM(idemem) + (r * headcountM(idemem)
  + *(maxindiv - headcountM(idemem)) / maxindiv)
IF (maxoffspring .gt. maxheadroom) maxoffspring = maxheadroom
END

SUBROUTINE SEX SORTING (numbmales, numbfemales, Male, Female, Sexm, idemem, HeadcountM, Migrant)

INCLUDE + [DREW.FORTRAN.EXPERIMENT]16LOCUS1PEAKCALL.FOR'
INTEGER Migrant (maxdeme, maxheadroom, maxchromo, maxlocus)
INTEGER Female (maxheadroom, maxchromo, maxlocus)
INTEGER Male (maxheadroom, maxchromo, maxlocus)
INTEGER SexM (maxdeme, maxheadroom)
INTEGER HeadcountM (maxdeme)
INTEGER numbmales
INTEGER numbfemales
INTEGER idemeM
INTEGER IndivM
INTEGER locusM
INTEGER ICOPYM

numbmales = 0
numbfemales = 0

DO 205 Indivm = 1, headcountM (idemem)
IF (SexM (idemM, indivM), eq. 1) THEN
  numbfemales = numbfemales + 1
  DO 215 locutm = 1, maxlocus
    DO 225 icopyM = 1, maxchromo
      Female (numbfemales, icopyM, locusM) =
      Migrant(idemM, indivM, icopyM, locusM)
    CONTINUE
  CONTINUE
ELSE
  numbmales = numbmales + 1
  DO 235 locutm = 1, maxlocus
    DO 245 icopyM = 1, maxchromo
      Male (numbmales, icopyM, locusM) = Migrant
      (idemM, indivM, icopyM, locusM)
  CONTINUE
245 CONTINUE
SUBROUTINE MATCHMAKER(idemeM, IDUM, IDUM2, ISEED, newind, numbmales, numbmales, Male, Female, Young, SexY, INDIVY)

INCLUDE + '[DREW.FORTRAN.EXPERIMENT]16LOCUS1PEAKCALL.FOR'

INTEGER belle
INTEGER beau
INTEGER IdemeY
INTEGER IndivY
INTEGER IcopyY
INTEGER locusY
INTEGER IdemeM
INTEGER IcopyM
INTEGER locusM
INTEGER newind
INTEGER numbmales
INTEGER numbmales
INTEGER male (maxheadroom, maxchromo, maxlocus)
INTEGER female (maxheadroom, maxchromo, maxlocus)
INTEGER SexY (maxdeme, maxheadroom)
INTEGER Young (maxdeme, maxheadroom, maxchromo, maxlocus)
belle = Ran2 (IDUM2) * numbmales + 1
beau = Ran2 (IDUM2) * numbmales + 1
idemeY = idemeM
DO 305 icopyY = 1, maxchromo
indivy = newind
IF (mod (icopyY, 2) .eq. 0) THEN
   DO 315 locusY = 1, maxlocus
      icopyM = (IRBIT2(ISEED) + 1)
      Young (idemeY, indivY, icopyY, locusY) =
         + female (belle, icopyM, locusM)
      CALL MUTATION (Young, IDUM, IDUM2, ISEED,
         + idemeY, indivY, icopyY, locusY)
   315 CONTINUE
ELSE
   DO 325 locusY = 1, maxlocus
      icopyM = (IRBIT2(ISEED) + 1)
      IF (icopyM .eq. 3) icopyM = 2
      Young (idemeY, indivY, icopyY, locusY) =
         + Male (beau, icopyM, locusM)
      CALL MUTATION (Young, IDUM, IDUM2, ISEED,
         + idemeY, indivY, icopyY, locusY)
   325 CONTINUE
END IF

305 CONTINUE
SexY (idemeY, indivY) = (IRBIT2(ISEED) + 1)
END

SUBROUTINE MUTATION (Young, IDUM, IDUM2, ISEED,
+ idemeY, indivY, icopyY, locusY)

* *********************************************************
* this subroutine changes the allelic value of the copy at a locus with a frequency of
* 1/mutation rate
* *********************************************************

INCLUDE + [DREW.FORTRAN.EXPERIMENT]16LOCUS1PEAKCALL.FOR'
INTEGER IdemeY
INTEGER IndivY
INTEGER IcopyY
INTEGER locusY
  INTEGER mute
  INTEGER Young(maxdeme, maxheadroom, maxchromo, maxlocus)

  mute = Ran1(IDUM) * mutationrate + 1
IF (mute .eq. mutationrate) THEN
  IF (Young(idemeY, indivY, IcopyY, locusY).eq. 1) THEN
    Young(idemeY, indivY, IcopyY, locusY) = 2
  ELSE
    IF (Young(idemeY, indivY, IcopyY, locusY).eq. 2)
+    Young(idemeY, indivY, IcopyY, locusY) = 1
ENDIF
ENDIF
END

SUBROUTINE STAT (igen, young, totind, headcountY, iskip, itrial)

INCLUDE + [DREW.FORTRAN.EXPERIMENT]16LOCUS1PEAKCALL.FOR'
INTEGER idemey
INTEGER indivy
INTEGER itrait
INTEGER totind
INTEGER popind
INTEGER IGEN
INTEGER ISKIP
INTEGER IIT
INTEGER IJT
INTEGER LOCUSY
INTEGER ICOPYY
INTEGER I
INTEGER ITRIAL
INTEGER HeadcountY(maxdeme)
INTEGER Young(maxdeme, maxheadroom, maxchromo, maxlocus)
INTEGER IndivDATA(0:(2*maxlocus-1))
REAL fitness
INTEGER hetadv
INTEGER delrec
INTEGER load
INTEGER Hetstart
INTEGER t
  INTEGER Delrecstart
REAL IP2
  INTEGER Iplace
REAL prob

prob = 0
    IF (mod (igen, iskip) .eq. 0) THEN
        DO 3000 idemey=1, maxdeme
        DO 3200 indivy=1, headcounty(idemey)
            DO 3250 j=1, (2*maxlocus)
                IndivDATA(j)=0
            CONTINUE
        Iplace = 0
        DO 3300 locusy=1, maxqlocus ! find the genotype
            DO 3400 icopyY = 1, maxchromo
                Iplace = + Iplace + Young(idemeY, indivY, icopyY, locusY)
            CONTINUE
            CONTINUE
        IP2=Iplace
        IF (IP2 .ge. peak) THEN
            prob = ((1.0-(((ip2-peak)/peak)*S))*L)
        ELSE
            prob = ((1.0-(((peak-ip2)/peak)*S))*L)
        ENDIF
    hetadv = 0
    delrec = 0
    Hetstart= (maxqlocus+1)
    Delrecstart= (maxhetadv+1) 

    DO 3100 locount = Hetstart, maxhetadv ! counts the impact of all het adv loci
       IF (YOUNG(idemey, indivy, 1, locount) .eq. + YOUNG(idemey, indivy, 2, locount)) THEN
           hetadv = hetadv + 1
       ELSE
           END IF
    3100 CONTINUE
    DO 3150 hicount = Delrecstart, maxlocus
       IF ((YOUNG(idemey, indivy, 1, hicount) .eq. 1) + .and. (YOUNG(idemey, indivy, 2, hicount).eq. 1)) THEN
           delrec = delrec + 1
       END IF
    3150 CONTINUE

    fitness = prob-(inbeffect*(delrec + hetadv))
    ia=-1
    DO 2017 ilocus=1, maxlocus
       DO 2018 icopy=1, maxchromo
           ia=ia+1
           IndivDATA(ia)= YOUNG(idemey, indivy, icopy, Ilocus)
    2018 CONTINUE
    DO 2017 ilocus=1, maxlocus
       DO 2018 icopy=1, maxchromo
           ia=ia+1
           IndivDATA(ia)= YOUNG(idemey, indivy, icopy, Ilocus)
    2018 CONTINUE
    max2=2*maxlocus-1
    Write (16,5000) (Itrial, Igen, idemey, indivy, fitness, + hetadv, delrec, (Indivdata(i), i=0,232))
SUBROUTINE STAT2 (igen,young,totind,headcounty,iskip,itrail)

INCLUDE '[DREW.FORTRAN.EXPERIMENT]16locus1peakcall.for'

INTEGER idemey
INTEGER indivy
INTEGER itrait
INTEGER totind
INTEGER popind
INTEGER IGEN
INTEGER ISKIP
INTEGER IIT
INTEGER IJT
INTEGER LocusY
INTEGER I
INTEGER ITRIAL
INTEGER HeadcountY(maxdeme)
INTEGER poptcum(0:maxqlocus)
INTEGER demetcum(0:maxqlocus)
INTEGER Young(maxdeme,maxheadroom,maxchromo,maxlocus)
REAL pf(0:maxqlocus)
REAL df(0:maxqlocus)
REAL Pwreal
REAL Dwreal

IF(mod(igen,iskip) .eq. 0)THEN
popind=0
Pwreal=0
Do 51 iit=0,maxqlocus
   poptcum(iit)=0
51 CONTINUE

Do 2000 idemey=1,maxdeme
dwreal=0.0
Do 52 ijt=0,maxqlocus
   demetcum(ijt)=0
52 CONTINUE

IF (headcountY(idemey) .eq. 0) GOTO 2000
DO 3333 indivy=1,headcounty(idemey)
itrait=0
   DO 3020 locusy=1,maxlocus
      DO 3030 icopyY=1,maxchromo
         IF(Young(idemey,indivY,icopyY,locusY).lt.1)THEN
            type *,'oh no'
            ELSE
           ENDIF
         Itrait=Ittrait+Young(idemey,indivy,icopyY,locusY)-1
3030 CONTINUE
3020 CONTINUE
Demetcum(itrait)=demetcum(itrait)+1
Poptcum(itrait)= poptcum(itrait) +1
popind=popind+1
3333 CONTINUE

DO 53 iit=0,maxqlocus
   DF(iit)=(demetcum(iit)/(1.0*headcountY(idemeY)))
   IF (iit .ge. peak)THEN
      Dwreal=Dwreal+(1.0-(((iit-peak)/peak)*S))*df(iit)
   ELSE
      Dwreal=Dwreal+(1.0-((peak-iit)/peak)*S))*df(iit)
   ENDIF
53 CONTINUE

* WRITE(15,4000)((df(i),i=0,10),dwreal,headcountY(idemeY),
* + iegen,idemeY,itrial)
2000 CONTINUE

Do 54 ijt=0,maxqlocus
   pf(ijt)=(poptcum(ijt)/(1.0*popind))
   IF (ijt .ge. peak)THEN
      Pwreal=Pwreal+(1.0-(((ijt-peak)/peak)*S))*pf(ijt)
   ELSE
      Pwreal=Pwreal+(1.0-((peak-ijt)/peak)*S))*pf(ijt)
   ENDIF
54 CONTINUE

* WRITE(14,4000)((pf(i),i=0,10),pwreal,popind,igen,itrial)
4000 FORMAT (1x,f6.4,1x,f6.4,1x,f6.4,1x,f6.4,1x,f6.4,1x,f6.4,1x,f6.4,1x,f6.4,1x
   + ,f6.4,1x,f6.4,1x,f6.4,1x,f6.4,1x,f6.4,3x,f10.6,2x,i5,2x,i3,2x,i3)
   ELSE
   ENDIF
   END

FUNCTION RAN1(IDUM)

*************************************************************************
* RETURNS A UNIFORM RANDOM DEVIATE BETWEEN 0.0 AND 1.0 SET IDUM TO ANY
* NEGATIVE VALUE TO INITIALIZE OR REINITIALIZE THE SEQUENCE.
* THIS ROUTINE IS TAKEN FROM NUMERICAL RECIPIES (FORTRAN) BY PRESS
* FLANNERY, TEUKOLSKY AND VETTERLING (CAMBRIDGE PRESS). THIS IS
* SLOWER THAN RAN2
*************************************************************************

REAL*8 R(97)
PARAMETER (M1=134456,IA1=8121, IC1=28411, RM1=1.0/M1)
PARAMETER (M2=243000,IA2=4561, IC2=51349, RM2=1.0/M2)
PARAMETER (M3=259200,IA3=7141, IC3=54773)
DATA IFF/0/ ! initialize on the first round

IF (IDUM .LT. 0 .OR. IFF .EQ. 0)THEN
IFF = 1
IX1=MOD(IC1-IDUM,M1)  ! SEED THE FIRST ROUND
IX1=MOD(IA1*IX1+IC1,M1)
IX2=MOD(IX1,M2)        ! USE IT TO SEED THE SECOND
IX1=MOD(IA1*IX1+IC1,M1)
IX3=MOD(IX1,M3)        ! AND THE THIRD

DO 11 J = 1, 97        ! FILL THE TABLE WITH UNIFORM DEVIATES
IX1=MOD(IA1*IX1+IC1,M1) ! provided by first two routines
IX2=MOD(IA2*IX2+IC2,M2)
R(J) = (FLOAT(IX1)+FLOAT(IX2)*RM2)*RM1 !Combine the low and high order pieces here
CONTINUE

IDUM = 1
ENDIF
IX1=MOD(IA1*IX1+IC1,M1) ! BEGIN NORMAL DRAWS
IX2=MOD(IA2*IX2+IC2,M2)
IX3=MOD(IA3*IX3+IC3,M3)
J=1+(97*IX3)/M3
IF(J .GT. 97 .OR. J .LT. 1) PAUSE
RAN1=R(J)
R(J) = (FLOAT(IX1)+FLOAT(IX2)*RM2)*RM1
RETURN
END

FUNCTION RAN2(IDUM2)
* *************************************************************
* RETURNS A UNIFORM RANDOM DEVIATE BETWEEN 0.0 AND 1.0 SET IDUM2 TO ANY
* NEGATIVE VALUE TO INITIALIZE OR REINITIALIZE THE SEQUENCE.
* THIS ROUTINE IS TAKEN FROM NUMERICAL RECIPES (FORTRAN) BY PRESS
* FLANNERY, TEUKOLSKY AND VETTERLING (CAMBRIDGE PRESS). THIS IS
* FASTER THAN RAN1 BUT IT GENERATES LESS RANDOM NUMBERS SO DO NOT RELY
* ON IT FOR NUMBERS THAT REQUIRE NEARLY ALL THE SIGNIFICANT BITS OF WORD
* SIZE
* AVAILABLE. SPECIFICALLY. I WOULD USE CAUTION BEYOND THE FOURTH
* SIGN.DIG.
* *************************************************************
*
INTEGER IR(97)
PARAMETER (M=714025, IA=1366, IC=150889, RM=1.0/M)
DATA IFF/0/   ! initialize on the first round

IF (IDUM2 .LT. 0 .OR. IFF .EQ. 0)THEN
IFF = 1
IDUM2=MOD(IC-IDUM2,M)  ! SEED THE FIRST ROUND

DO 11 J = 1, 97        ! FILL THE TABLE WITH UNIFORM DEVIATES
IDUM2=MOD(IA*IDUM2+IC,M)
IR(J) = IDUM2
CONTINUE

IDUM2=MOD(IA*IDUM2+IC,M)
IY = IDUM2
ENDIF

J=1+(97*IY)/M        ! BEGIN HERE EXCEPT ON INITIALIZATION
IF(J .GT. 97 .OR. J .LT. 1) PAUSE
IY=IR(J)
RAN2=IY*RM
IDUM2=MOD(IA*IDUM2+IC,M)
IR(J)=IDUM2

RETURN
END

FUNCTION IRBIT2(ISEED)
* *-----------------------------------------------------------------
* RETURNS A RANDOM BIT AS AN INTEGER (1, OR 0), BASED ON THE 30 LOW
* SIGNIFICANCE BITS IN ISEED. ISEED SHOULD BE SET TO A LARGE NEGATIVE
* INTEGER TO INITIALIZE OR REINITIALIZE THE SEQUENCE.
* THIS ROUTINE IS TAKEN FROM NUMERICAL RECIPIES (FORTRAN) BY PRESS,
* FLANNERY, TEUKOLSKY AND VETTERLING (CAMBRIDGE PRESS).
* NOTE THAT THIS IS FUNCTION IS NOT WRITEN IN FORTRAN 77, IT USES THE
* VAX-11 FUNCTIONS IXOR (BITWISE EXCLUSIVE OR) ISHFT (BITWISE SHIFT) AND
* IAND (BITWISE AND) I WOULD EXPECT THAT THESE ARE PRETTY
* COMMON EXTENTIONS HOWEVER.
* *-----------------------------------------------------------------

PARAMETER(ib1=1,ib4=8,ib6=32,ib30=536870912,mask=ib1+ib4+ib6)
IF (IAND(Iseed, ib30) .NE. 0) THEN ! change all the masked bits, shift, and put 1 into bit 1.
    iseed=IOR(ISSHFT(IIEOR(Iseed,mask), 1),IB1)
    irbit2=1
ELSE
    iseed=IAND(ISSHFT(Iseed,1),NOT(IB1))
    irbit2=0
ENDIF
RETURN
*
    type *, irbit is equal to ',irbit2
END
APPENDIX B

SOURCE CODE FOR CAPTIVE RUNS

The following program is the simulation written in FORTRAN that utilized the associated parameter file to test captive and demic structured populations during the simulations. This file was augmented to test for non-structured populations as well.

* PHASE II final PHASE!!!!!!!
* Title: Maintenance of Variation During Captive Breeding in a Finite Number of Demes

* Version 1.3.final now 21.99
* This program is intended to simulate the change in gene frequency of multiple demes under selective pressure at a number of epistatic loci.
* Small portions of this program were borrowed from a program entitled "Gametophytic Incompatibility" which was written by Kieth Crandall
* (Jan. 5th, 1987). The function ran2 was acquired from B.S. Wier's Genetic Data Analysis, 1990. Sinaeur Assoc. Ran2 was modified by F Moore June 1991.
* swright, rodeo and stat have been augment and perfected by Andrew Jones
* all other routines have been tweaked

* Author: Drew Jones & Francisco Moore, JAN 28 2014
* Version without inbred fubar ***Drew

* Declarations:

INCLUDE '[drew.fortran.experiment]calldev.FOR'
* The following arrays are used to store genotypes by deme, individual, locus, and chromosome (Genotypic arrays for each stage).
INTEGER Migrant (maxdeme,maxheadroom,maxchromo,maxlocus)
INTEGER Adolescent (maxdeme,maxheadroom,maxchromo,maxlocus)
INTEGER Young (maxdeme,maxheadroom,maxchromo,maxlocus)

* The following arrays are used to store a sex by deme, and individual (sex arrays for each stage).
INTEGER SexY (maxdeme, maxheadroom)
INTEGER SexA (maxdeme, maxheadroom)
INTEGER SexM (maxdeme, maxheadroom)

* The following arrays are used to keep track of the number of individuals within a deme
INTEGER HeadcountY (maxdeme)
INTEGER HeadcountA (maxdeme)
INTEGER HeadcountM (maxdeme)
REAL peakht (maxdeme)
The following variables will be used to keep track of cells of the different arrays, throughout the program.

- INTEGER IdemeY
- INTEGER IdemeM
- INTEGER IdemeA
- Totind is a variable which counts the total number of individuals which have been produced in all demes over all generations
- INTEGER totind

The following variables control loops
- INTEGER igen
- INTEGER iskip
- INTEGER genwrt
- INTEGER ITRIAL
- ProbG is the probability of migrating a distance using a gamma distribution
  - REAL ProbG (0:maxXdist)

The following arrays represent numbers of demes at a distance
- INTEGER DX (0:xmax, maxnumb)
- INTEGER DY (0:xmax, maxnumb)
- INTEGER Numb (0:xmax)

PARAMETER (iskip =10)

Program execution begins at this point

*******************************************************************************
IDUM seeds the random number generator and should be a large negative number.
The value for IDUM could be read off the clock.
secnds is a VAX specific command that returns the clock reading in seconds which is here
used as a seed value if this command is not available a large negative number should be
used as a seed.
*******************************************************************************

IDUM = -1 * secnds(0.00)*100  !-3894182
IDUM2 = IDUM
ISEED = IDUM

Open Files Routine
this is a routine which opens output files for storage of data
and then writes the headers for the files

OPEN (unit=11,file=
  'drew.fortran.experiment]popdeme1.dat',
  status='old',access='sequential',form='formatted',recl=2048)
OPEN (unit=12,file=
  'drew.fortran.experiment]popdeme2.dat',
  status='old',access='sequential',form='formatted',recl=2048)
OPEN (unit=13,file=
  'drew.fortran.experiment]popdeme3.dat',
  status='old',access='sequential',form='formatted',recl=2048)
OPEN (unit=14,file=
  'drew.fortran.experiment]popdeme4.dat',

OPEN (unit=15, file='[drew.fortran.experiment]popdeme5.dat', status='old', access='sequential', form='formatted', recl=2048)
OPEN (unit=16, file='[drew.fortran.experiment]popdeme6.dat', status='old', access='sequential', form='formatted', recl=2048)
OPEN (unit=17, file='[drew.fortran.experiment]popdeme7.dat', status='old', access='sequential', form='formatted', recl=2048)
OPEN (unit=18, file='[drew.fortran.experiment]popdeme8.dat', status='old', access='sequential', form='formatted', recl=2048)
OPEN (unit=19, file='[drew.fortran.experiment]popdeme9.dat', status='old', access='sequential', form='formatted', recl=2048)
OPEN (unit=20, file='[drew.fortran.experiment]popdeme10.dat', status='old', access='sequential', form='formatted', recl=2048)
OPEN (unit=21, file='[drew.fortran.experiment]popdeme11.dat', status='old', access='sequential', form='formatted', recl=2048)
OPEN (unit=22, file='[drew.fortran.experiment]popdeme12.dat', status='old', access='sequential', form='formatted', recl=2048)
OPEN (unit=23, file='[drew.fortran.experiment]popdeme13.dat', status='old', access='sequential', form='formatted', recl=2048)
OPEN (unit=24, file='[drew.fortran.experiment]popdeme14.dat', status='old', access='sequential', form='formatted', recl=2048)
OPEN (unit=25, file='[drew.fortran.experiment]popdeme15.dat', status='old', access='sequential', form='formatted', recl=2048)
OPEN (unit=26, file='[drew.fortran.experiment]popdeme16.dat', status='old', access='sequential', form='formatted', recl=2048)
OPEN (unit=27, file='[drew.fortran.experiment]popdeme17.dat', status='old', access='sequential', form='formatted', recl=2048)
OPEN (unit=28, file='[drew.fortran.experiment]popdeme18.dat', status='old', access='sequential', form='formatted', recl=2048)
OPEN (unit=29, file='[drew.fortran.experiment]popdeme19.dat', status='old', access='sequential', form='formatted', recl=2048)
OPEN (unit=30, file='[drew.fortran.experiment]popdeme20.dat', status='old', access='sequential', form='formatted', recl=2048)
OPEN (unit=31, file='[drew.fortran.experiment]poprand1.dat', status='old', access='sequential', form='formatted', recl=2048)
OPEN (unit=32, file='[drew.fortran.experiment]poprand2.dat', status='old', access='sequential', form='formatted', recl=2048)
OPEN (unit=33, file=...
+ '__file__=\'[drew.fortran.experiment]poprand3.dat',
+ status='old',access='sequential',form='formatted',recl=2048)
OPEN (unit=34,file=
+ '__file__=\'[drew.fortran.experiment]poprand4.dat',
+ status='old',access='sequential',form='formatted',recl=2048)
OPEN (unit=35,file=
+ '__file__=\'[drew.fortran.experiment]poprand5.dat',
+ status='old',access='sequential',form='formatted',recl=2048)
OPEN (unit=36,file=
+ '__file__=\'[drew.fortran.experiment]poprand6.dat',
+ status='old',access='sequential',form='formatted',recl=2048)
OPEN (unit=37,file=
+ '__file__=\'[drew.fortran.experiment]poprand7.dat',
+ status='old',access='sequential',form='formatted',recl=2048)
OPEN (unit=38,file=
+ '__file__=\'[drew.fortran.experiment]poprand8.dat',
+ status='old',access='sequential',form='formatted',recl=2048)
OPEN (unit=39,file=
+ '__file__=\'[drew.fortran.experiment]poprand9.dat',
+ status='old',access='sequential',form='formatted',recl=2048)
OPEN (unit=40,file=
+ '__file__=\'[drew.fortran.experiment]poprand10.dat',
+ status='old',access='sequential',form='formatted',recl=2048)
OPEN (unit=41,file=
+ '__file__=\'[drew.fortran.experiment]poprand11.dat',
+ status='old',access='sequential',form='formatted',recl=2048)
OPEN (unit=42,file=
+ '__file__=\'[drew.fortran.experiment]poprand12.dat',
+ status='old',access='sequential',form='formatted',recl=2048)
OPEN (unit=43,file=
+ '__file__=\'[drew.fortran.experiment]poprand13.dat',
+ status='old',access='sequential',form='formatted',recl=2048)
OPEN (unit=44,file=
+ '__file__=\'[drew.fortran.experiment]poprand14.dat',
+ status='old',access='sequential',form='formatted',recl=2048)
OPEN (unit=45,file=
+ '__file__=\'[drew.fortran.experiment]poprand15.dat',
+ status='old',access='sequential',form='formatted',recl=2048)
OPEN (unit=46,file=
+ '__file__=\'[drew.fortran.experiment]poprand16.dat',
+ status='old',access='sequential',form='formatted',recl=2048)
OPEN (unit=47,file=
+ '__file__=\'[drew.fortran.experiment]poprand17.dat',
+ status='old',access='sequential',form='formatted',recl=2048)
OPEN (unit=48,file=
+ '__file__=\'[drew.fortran.experiment]poprand18.dat',
+ status='old',access='sequential',form='formatted',recl=2048)
OPEN (unit=49,file=
+ '__file__=\'[drew.fortran.experiment]poprand19.dat',
+ status='old',access='sequential',form='formatted',recl=2048)
OPEN (unit=50,file=
+ '__file__=\'[drew.fortran.experiment]poprand20.dat',
+ status='old',access='sequential',form='formatted',recl=2048)

OPEN (unit=9,file=
OPEN (unit=10, file=+
[drew.fortran.experiment]4dv_dem.csv',
+ status='new', access='sequential', form='formatted', recl=2048)

OPEN (unit=60, file=+
[drew.fortran.experiment]ioffv_q.nex',
+ status='new', access='sequential', form='formatted', recl=2048)

OPEN (unit=61, file=+
[drew.fortran.experiment]ioffv_inb.nex',
+ status='new', access='sequential', form='formatted', recl=2048)

WRITE(61,2000)
WRITE(60,2000)
WRITE(9,1005)(1-P),maxqlocus,maxinbred,k,s,l
WRITE(10,1005)(1-P),maxqlocus,maxinbred,k,s,l
WRITE(9,1006)
WRITE(10,1007)

1004 FORMAT(3x,'z=0',4x,'z=1',4x,'z=2',4x,'z=3',4x,'z=4',4x,'z=5',
+ 4x,'z=6',4x,'z=7',4x,'z=8',4x,'z=9',4x,'z=10',4x,'preal',3x,
+ '# indiv',5x,'gen',3x,'itrial')

1006 FORMAT(2x,'trial',4x,'#indiv',1x,'#demes',
+ 2x,'#pop avg fit',1x,'#pop var fit',1x,'#btw var fit',
+ 1x,'#pop avg q',1x,'#pop var q',1x,'#btw var q',
+ 1x,'#pop avg het',1x,'#pop var het',1x,'#btw var het',
+ 1x,'#pop avg del',1x,'#pop var del',1x,'#btw var del',
+ 1x,'#pop avg inb',1x,'#iseed',10x,'#pop',10x,'#breed',10x,'zoosd',
+ 'program name',10x,'qsurf')

1005 FORMAT('7-15-12 ','migration=',f10.8,1x,'#Quantitative loci=','
+ ',i3,'#inbreeding loci= ',i3,1x,k=' ',f6.4,1x,s=' ',f6.4,1x,
+ 'l= ',f6.4)

1007 FORMAT(2x,'trial',4x,'#deme',2x,'#pop',10x,'AVG Fit',
+ 1x,'#var fit',1x,'#dem het',1x,'#var het',
+ 1x,'#dem del',1x,'#var del',1x,'#dem q',
+ 1x,'#var q',4x,'#peak ht',1x,'#iseed',10x,'#pop',
+ 6x,'#breed',10x,'zoosd',10x,'program name',10x,'qsurf')

2000 FORMAT('#NEXUS',/,'Begin gadata;')
* Here is a good spot to define the probabilities of Gamma Function probg

CALL GAMMA MIGRATION (probg)

* Now the number of demes which are located a given distance away from a given
deme need to determined for use during periods of migration.

CALL NUMBDIST (Numb, DX, DY)

* loop through #? trials

DO 886 itrial = 1, 20 !40 no longer running randomly drawn simulations
igen = 0

* Now it is time too seed the parental Genotypes, and initialize the adolescent
Genotypes

CALL SWRIGHT (igen, iseed, peakht) makes the wrightian fitness surface for each deme

CALL PARSEED (Sexy, Young, Sexa, Adolescent, HeadcountA, HeadcountY,
totind, IDUM, IDUM2, ISEED, itrial)

* Now, the statistics on the parental generation can be done (see flow chart C)

CALL STAT (igen, young, totind, headcountY, iskip, itrial, peakht,
+ idum, itrial)

* Now it is time to begin looping through the generations

DO 100 igen = 1, maxgen

CALL SWRIGHT (igen, iseed, peakht)

* The youngsters now go through a bout of selection.

CALL SELECTION (Adolescent, Young, HeadcountY, HeadcountA, SexY,
+ SexA, IDUM, IDUM2, ISEED, peakht)

* It is time to Initialize (re-initialize) the migrant demes so they will be ready
for new migrants.

CALL INITIALIZE MIGRANT (Migrant, HeadcountM, SexM)
Now that we have initialized the migrant array migration can take place
CALL MIGRATION (SexM, SexA, Adolescent, Migrant, HeadcountM, IDUM, IDUM2, ISEED, Numb, DX, DY, ProbA, HeadcountA, igen)

Now the young and Adolescent Arrays need to be reset
CALL INITIALIZE Y AND A (Young, Adolescent, HeadcountY, HeadcountA, SexY, SexA)

Mating time
CALL MATING (Migrant, Young, SexM, SexY, totind, HeadcountM, HeadcountY, IDUM, IDUM2, ISEED, igen)

CALL STAT (igen, young, totind, headcountY, iskip, itrial, peakht, idum, itrial)

100 CONTINUE
CLOSE(itrial+10)

886 CONTINUE
END

SUBROUTINE SWRIGHT (igen, iseed, peakht)
INCLUDE + '[drew.fortran.experiment]calldv.FOR'

INTEGER igen
INTEGER idemey
REAL PEAKHT(maxdeme)

* sel = 0 for variable environment...1 for stable environment

IF(igen.lt.20)THEN
DO 9877 idemey=1,maxdeme
peakht(idemey)=24 !should only be for captive generations
9877 CONTINUE
ELSE IF(igen.eq.21)THEN
DO 9875 idemey=1,maxdeme
peakht(idemey)=(RAN2(iseed)*(2*maxqlocus))
9875 CONTINUE
ELSE

66
IF((sel.eq.0).and.(igen.gt.20))THEN
DO 9876 idemey=1,maxdeme
peakht(idemeY)= peakht(idemeY)+(RAN2(iseed)*8)-4
IF(peakht(idemeY).lt.0) peakht(idemeY)=0
IF(peakht(idemeY).gt.32) peakht(idemeY)=32
9876 CONTINUE
ELSE
END IF
END IF
END

SUBROUTINE PARSEED (Sexy,Young,Sexa,Adolescent,HeadcountA,
  + HeadcountY,totind,IDUM,IDUM2,ISEED,itrial)
INCLUDE
+ '[drew.fortran.experiment]calldv.FOR'
* ******************************************************
* The following variables and parameters are the same as in the main program
* ******************************************************
INTEGER Adolescent (maxdeme,maxheadroom,maxchromo,maxlocus)
INTEGER YounG(maxdeme,maxheadroom,maxchromo,maxlocus)
INTEGER SEXY (maxdeme, maxheadroom)
INTEGER SEXA (maxdeme, maxheadroom)
INTEGER zoogen(0:(2*maxlocus))
INTEGER j
INTEGER ia
INTEGER Headcounty (maxdeme)
INTEGER Headcounta (maxdeme)
INTEGER Idemey
INTEGER Indivy
INTEGER Icopyy
INTEGER locusy
INTEGER Idemes
INTEGER Indivs
INTEGER Idemea
INTEGER Icopya
INTEGER Indiva
INTEGER Locusa
INTEGER totind
INTEGER IOREO1
INTEGER IOREO2
INTEGER IOREO3
INTEGER IOREO4
INTEGER hcount
INTEGER startcount
INTEGER filenum

* ******************************************************
* This Subroutine seeds the genotypes and sexes of the parental generation
* see flow chart B
* ******************************************************
The first thing to do is set the population counters to zero

totind = 0
filenum=(itrial+10)

Now loop through and create the proper number of individuals in each
population and flag all non-existent individuals with -1 for young and
adolescents

DO 20 idemeY=1,maxdeme
    idemea = idemey
    headcountA (idemeA) = 0
    headcountY(idemeY) = 0

DO 30 IndivY = 1,maxheadroom !loop through all individuals
    ia = -1
    indivA = indivY
    SexA (idemeA,indivA) = -1 ! adolescents don't exist yet
    IF (indivY.le.(maxzooheadroom).and.(idemeY.le.maxzoodeme)) THEN ! we only have
        ! individuals in half of the potential spots
            READ(filenum,1111)idemes,indivs,sexY(idemeY,indivY),
            (zoogen(j),j=0,231)
        DO 40 locusY = 1,maxlocus ! we need to seed values at each copy
            locusa = locusY
            DO 50 icopyY = 1,maxchromo ! and each locus
                icopyA = icopyY
                ia=ia+1
                Young(idemeY,indivY,icopyY,locusY)=zoogen(ia)
                Adolescent (idemeA,indivA,icopyA,locusa) = -1 ! adolescents don't exist yet
            END DO 50 icopyY
        CONTINUE 40
        HeadcountY (idemeY) = HeadcountY (idemeY) + 1 ! a new individual was just created
        totind = totind + 1
    ELSE ! these youngsters don't exist yet
        SexY (idemeY,indivY) = -1 ! -1 indicates that the individual is non-existent
        DO 60 icopyY = 1,maxchromo
            icopyA = icopyY
            DO 70 locusY = 1,maxlocus
                locusa = locusY
                Young (idemeY,indivY,icopyY,locusY) = -1 ! these youngsters don't exist yet
        Adolescent (idemeA,indivA,icopyA,locusa) = -1! adolescents don't exist yet
    END IF 30
    CONTINUE 20
1111 FORMAT(I5,,,I5,,,I2,,,232(1X,I2))
SUBROUTINE NUMBDIST(Numb, DX, DY)
*
************************************************************************************
*  Now the number of demes which are located a given distance away from a given    *
*  deme need be determined for use during periods of migration. This subprogram stores *
*  the number (Numb(ir)) of demes which are a given distance (ir) away from any deme. It *
*  also stores the coordinates (DX(ir,Numb(ir)),DY(ir,Numb(ir))) of the different demes  *
*  relative to the originating deme (Deme of origin = (0,0)).                          *
************************************************************************************
*  The following variables represent distances and what not which are used         *
*  in calculating the number of demes at a distance                                *
*
INTEGER ir
INTEGER a
INTEGER b
REAL*8  c
REAL*8  d
INTEGER i
INTEGER j

INCLUDE + '[drew.fortran.experiment]calldv.FOR'
*
The following variables and parameters are the same as in the main program
  INTEGER Dx (0:xmax, maxnumb)
  INTEGER Dy (0:xmax, maxnumb)
  INTEGER numb (0: xmax)

DO 100 i=0,XMAX
    Numb(i) = 0
DO 200 J=1, MAXNUMB
    DX(i, J) = 0
    DY(i, J) = 0
200   CONTINUE
100   CONTINUE

IF (MOD(xmax,2) .NE. 0) THEN  !if xmax is odd -maxxdist to maxxdist won't overlap
  DO 120 i = -maxXdist, maxXdist  ! loop through all demes
     DO 130 j = -maxYdist, maxYdist ! in all directions
        a = i    ! calculate the integer distance to this deme
        b = j
        c = ((a*a) + (b*b))
        d = SQRT (c)
        ir = INT (d) ! calculate the integer distance to this deme
        IF (ir .gt. maxXdist) ir = maxXdist ! anything greater than a half of the way around the
*        torus is equally far
        Numb(ir) = Numb(ir) + 1     ! One more at this distance
        DX(ir, Numb(ir)) = i      !assign these coordinates to this number deme
        DY(ir, Numb(ir)) = j
130   CONTINUE
120   CONTINUE
ELSE ! if Ymax is even you don't want to overlap map edges
  DO 140 i = -maxXdist, maxXdist ! loop through all demes
  DO 150 j = -maxYdist, maxYdist-1 ! in all directions
    a = i ! calculate the integer distance to this deme
    b = j !
    c = ((a*a) + (b*b)) !
    d = SQRT (c) !
    ir = INT (d) ! calculate the integer distance to this deme
    IF (ir .gt. maxXdist) ir = maxXdist ! anything greater than a half of the way around the * torus is equally far
      Numb(ir) = Numb(ir) + 1 ! One more at this distance
      DX(ir, Numb(ir)) = i !assign these coordinates to this number deme
      DY(ir, Numb(ir)) = j
  150 CONTINUE
  140 CONTINUE
END IF
ELSE ! if Xmax is even you don't want to overlap map edges
  IF (MOD(ymax,2).NE.0) THEN !if ymax is odd -maxydist to maxydist won't overlap
    DO 160 i = -maxXdist, maxXdist-1 ! loop through all demes
    DO 170 j = -maxYdist, maxYdist ! in all directions
      a = i ! calculate the integer distance to this deme
      b = j !
      c = ((a*a) + (b*b)) !
      d = SQRT (c) !
      ir = INT (d) ! calculate the integer distance to this deme
      IF (ir .gt. maxXdist) ir = maxXdist ! anything greater than a half of the way around the * torus is equally far
        Numb(ir) = Numb(ir) + 1 ! One more at this distance
        DX(ir, Numb(ir)) = i !assign these coordinates to this number deme
        DY(ir, Numb(ir)) = j
    170 CONTINUE
    160 CONTINUE
  ELSE ! if Ymax is also even you don't want to overlap map edges
    DO 180 i = -maxXdist, maxXdist-1 ! loop through all demes
    DO 190 j = -maxYdist, maxYdist-1 ! in all directions
      a = i ! calculate the integer distance to this deme
      b = j !
      c = ((a*a) + (b*b)) !
      d = SQRT (c) !
      ir = INT (d) ! calculate the integer distance to this deme
      IF (ir .gt. maxXdist) ir = maxXdist ! anything greater than a half of the way around the * torus is equally far
        Numb(ir) = Numb(ir) + 1 ! One more at this distance
        DX(ir, Numb(ir)) = i !assign these coordinates to this number deme
        DY(ir, Numb(ir)) = j
  190 CONTINUE
  180 CONTINUE
END IF
END

SUBROUTINE SELECTION (Adolescent, Young, HeadcountY, HeadcountA, + SexY, SexA, IDUM, IDUM2, ISEED, peakht)
INTEGER Adolescent (maxdeme,maxheadroom,maxchromo,maxlocus)
INTEGER Young (maxdeme,maxheadroom,maxchromo,maxlocus)
INTEGER SEXY (maxdeme,maxheadroom)
INTEGER SEXA (maxdeme,maxheadroom)
INTEGER Headcounty (maxdeme)
INTEGER Headcounta (maxdeme)
INTEGER Idemey
INTEGER Indivy
REAL peakht(maxdeme)

***********

These two variables are used to compare the probability of survival and chance

**********

REAL prob
REAL killer

In this subroutine individuals in the young array are exposed to genotype dependent selection. The selection probabilities are as follows: If a genotype has at least one dominant allele (2) at each locus then the probability of survival is \((1 + K*S) \times L\), \(L\) is the survival probability of a doubly homozygous recessive, and all individuals with at least one dominant allele at only one of the two loci experiences a prob of \((1-S)*L\). Those that live through selection will be placed in the same deme in the adolescent array.

DO 220 Idemey = 1, maxdeme !loop through all demes
   DO 230 Indivy = 1, HeadcountY (idemey) !loop through all individuals
      CALL SURVIVAL PROBABILITY (prob, Young, idemey, indivy,
      + peakht)
      killer = RAN2(IDUM2)
      If (killer .le. prob) CALL ADOLESCENT STORAGE (Young,
      + Adolescent, SexY, SexA, idemey, indivy, headcountA)
230   CONTINUE
220  CONTINUE
END

SUBROUTINE SURVIVAL PROBABILITY (prob, Young, idemey, indivy,
+ peakht)

INCLUDE + '[drew.fortran.experiment]calldv.FOR'

This subroutine calculates the genotype dependent survival probability of individuals, and is called by the selection subroutine.

INTEGER Iplace
REAL IP2
REAL Prob
Real Load

The following variables and parameters are the same as in the main program

INTEGER Young(maxdeme,maxheadroom,maxchromo,maxlocus)
INTEGER Idemey
INTEGER Indivy
INTEGER Icopyy
INTEGER locusy
REAL peakht(maxdeme)
REAL pk
REAL M
REAL B
REAL minfit
REAL maxfit
REAL peakmin
REAL peakmax

pk= peakht(idemeY)
minfit=((1*L)-S)
maxfit=(1*L)
prob= 0
Iplace = 0
peakmin = 0
peakmax = (2*maxqlocus)

DO 320 locusY = 1, maxqlocus          !find the genotype
    DO 330 icopyY = 1, maxchromo
        Iplace=Iplace+Young(idemeY, indivY, icopyY, locusY)-1
    330 CONTINUE
320 CONTINUE
IP2=Iplace

IF (Ip2 .lt. pk)THEN
    M = ((maxfit-minfit)/(peakht(idemeY)-peakmin))
    prob= ((M*IP2)+minfit)
ELSE IF (IP2.eq.peak)THEN
    prob= maxfit
else
    M = ((minfit-maxfit)/(peakmax-peakht(idemeY)))
    prob= ((M*(IP2-peakht(idemeY)))+maxfit)
endif

Call INBREEDING LOAD (IP2, Young, Idemey, Indivy, Load)
Prob=Prob-Load
END

Subroutine INBREEDING LOAD (IP2, Young, Idemey, Indivy, Load)

INCLUDE
+ '[drew.fortran.experiment]calldv.FOR'
* ***************************************************************************
* This subroutine calculates the individuals inbreeding load across all heterosis and deleterious
* recessive loci.
* It is called by the Survival Prob subroutine.
* ***************************************************************************

INTEGER IP2
REAL Load
INTEGER locount
INTEGER hetadv
INTEGER delrec
INTEGER Idemey
INTEGER hicount
INTEGER Indivy
INTEGER Hetstart
INTEGER Delrecstart
INTEGER Young (maxdeme,maxheadroom,maxchromo,maxlocus)
  INTEGER sumtotal
  hetadv = 0
  delrec = 0
load = 1.01
  Hetstart= (maxqlocus+1)
  Delrecstart= (maxhetadv+1)
  DO 301 locount = Hetstart, maxhetadv !counts the impact of all het adv loci
   IF (YOUNG(idemey,indivy,1,locount).eq. +
       YOUNG(idemey,indivy,2,locount)) THEN
     hetadv = hetadv + 1
   ELSE
     ENDIF
  301 CONTINUE
  DO 302 hicount = Delrecstart,maxlocus
   IF ((YOUNG(idemey, indivy, 1, hicount) .eq. 1) +
       .and. (YOUNG(idemey, indivy, 2, hicount).eq. 1)) THEN
     delrec = delrec + 1
   END IF
  302 CONTINUE
  sumtotal = (delrec + hetadv)
  Load = (inbeffect * sumtotal)
END

SUBROUTINE ADOLESCENT STORAGE (Young, Adolescent, SexY, SexA, +
  idemeY, indivY, headcountA)

*  *****************************************************************
*  this subroutine stores the young which has most recently made it through a bout of selection
*  into the adolescent array
*  *****************************************************************

INCLUDE +  '[drew.fortran.experiment]calldv.FOR'
  INTEGER Adolescent (maxdeme,maxheadroom,maxchromo,maxlocus)
  INTEGER Young (maxdeme,maxheadroom,maxchromo,maxlocus)
  INTEGER SEXY (maxdeme, maxheadroom)
  INTEGER SEA (maxdeme, maxheadroom)
  INTEGER Headcounta (maxdeme)
  INTEGER Idemey
  INTEGER Indivy
  INTEGER Icopyy
  INTEGER locusy
INTEGER Idemea
INTEGER Icopya
INTEGER Indiva
INTEGER Locusa

idemeA = idemeY  ! the deme of the individual does not change
headcountA(idemea) = headcountA(idemea) + 1
indivA = headcountA(idemea)
SexA (idemeA, indivA) = SexY (idemey, indivy)
DO 420 locusY = 1, maxlocus  ! loop through each locus
   locusA = locusY
   DO 430 icopyY = 1, maxchromo  ! and each chromosome
      icopyA = icopyY
      Adolescent (idemeA, indivA, icopyA, locusA) = Young  ! and copy the values
      (idemeY, indivY, icopyY, locusY)
430 CONTINUE
420 CONTINUE
END

SUBROUTINE INITIALIZE MIGRANT (Migrant, HeadcountM, SexM)

* ******************************************************
* this subroutine initializes the migrant array with null values so that migration between
* adolescent demes may begin
* ******************************************************

INCLUDE + '[drew.fortran.experiment]calldv.FOR'
INTEGER Migrant (maxdeme,maxheadroom,maxchromo,maxlocus)
INTEGER SEXM (maxdeme, maxheadroom)
INTEGER Headcountm (maxdeme)
INTEGER Idemem
INTEGER Indvm
INTEGER Icopym
INTEGER Locusm

DO 520 idemeM = 1, maxdeme  ! within each deme
   HeadcountM (idemeM) = 0
520 CONTINUE

DO 530 indivM = 1, maxheadroom  ! for each indiv
   Sexm (idemeM, indivM) = -1  ! initialize the sex
   DO 540 icopyM = 1, maxchromo  ! and for each copy
      Adolescent (idemeM, indivM, icopyM, locusM) = Young  ! and at each locus
      Migrant (idemeM, indivM, icopyM, locusM) = -1  ! initialize the genotype
   540 CONTINUE
530 CONTINUE

SUBROUTINE MIGRATION (SexM, SexA, Adolescent, Migrant, headcountM, +
                      IDUM,IDUM2,ISEED, Numb, DX, DY, ProbG, HeadcountA,igen)

* ******************************************************
* this subprogram controls the random migration of adolescent individuals between arrays
* to produce the migrant array of individuals.
* ******************************************************
INCLUDE
+ ['drew.fortran.experiment|calldv.FOR'
    INTEGER Migrant (maxdeme,maxheadroom,maxchromo,maxlocus)
    INTEGER Adolescent (maxdeme,maxheadroom,maxchromo,maxlocus)
    INTEGER SEXA (maxdeme, maxheadroom)
    INTEGER SEXM (maxdeme, maxheadroom)
    INTEGER Headcounta (maxdeme)
    INTEGER Headcountm (maxdeme)
    INTEGER Idemem
    INTEGER Idemea
    INTEGER Indiva
    INTEGER IKEEP
    INTEGER IKEEPD
    INTEGER IDIST
    INTEGER counter
    INTEGER Dx (0:xmax, maxnumb)
    INTEGER Dy (0:xmax, maxnumb)
    INTEGER numb (0:xmax)
    REAL ProbG (0:maxXdist)
    REAL RDIST
    indiva = (RAN2(IDUM2) * maxheadroom) + 1 !start with a random individual
    DO 620 Ikeep = 1, maxheadroom !loop through all possible individuals
        idemeA = (RAN2(IDUM2) * maxdeme) + 1 !start with a random deme
        DO 630 Ikeepd = 1, maxdeme !loop through all the possible demes
            IF ((Adolescent(idemeA, indivA, 1,1),NE. -1).AND.
                (Adolescent(idemeA, indivA, 1,1),NE. 0)) THEN
                counter=Ikeep
                CALL DEME CHOICE (idemeA,idemeM, Numb, DX, DY,
                IDUM,IDUM2,ISEED,ProbG,idist,rdist,igen,counter)
                !Choose a deme to migrate to
            END IF
            IF ((HeadcountM (idemeM)) + 1 .le. maxheadroom) !make sure there is room
                CALL MIGRANT STORAGE (Migrant, Adolescent, SexM,rdist, !if there is room
                store migrant
                + SexA, headcountM, headcountA, idemeA, indivA, idemeM,idist)
                ELSE
                    END IF
                    idemeA = idemeA + 1
                    IF (idemeA .gt. maxdeme) idemeA = 1
            630 CONTINUE
                    indivA = indivA + 1
                    * IF(igen.eq.1)THEN
                      * IF (indivA .GE. headcountA(idemeA)) indivA = 1
                      * ELSE
                          IF (indivA .GE. maxheadroom) indivA = 1
                      END IF
                    620 CONTINUE
                    END

            SUBROUTINE DEME CHOICE (idemeA, idemeM, Numb, Dx, DY, IDUM,
            + IDUM2,ISEED,ProbG,idist, rdist,igen,counter)
            *
            * CHOOSE A DEME AT A DISTANCE
* INCLUDE
+ ['drew.fortran.experiment]calldv.FOR'
  INTEGER Idemem
  INTEGER Idemea
  INTEGER IX
  INTEGER IY
  INTEGER Y0
  INTEGER X0
  INTEGER NRAND
  INTEGER IDIST
  INTEGER Dx (0:xmax, maxnumb)
  INTEGER Dy (0:xmax, maxnumb)
  INTEGER numb (0:xmax)
  INTEGER counter
  REAL RDIST
  REAL ProbG (0:maxXdist)

* THIS CODE DETERMINES THE X AND Y COORDINATES OF THE DEME FROM ITS DEME NUMBER
* ****************************************************************
  IF (MOD (idemeA, Xmax).eq. 0) THEN
    Y0 = (idemea/xmax)
  ELSE
    Y0 = (idemeA/Xmax) + 1
  END IF
  X0 = idemeA - ((Y0 - 1) * xmax)

* THIS CODE PICKS A RANDOM DISTANCE BASED ON THE PROB. FUNCTION
* GENERATED BY THE GAMMA FUNCTION ROUTINE
* ****************************************************************
  Rdist = RAN1(IDUM)
  DO 720 I = 0, maxXdist
    IF (Rdist .le. ProbG(I)) THEN
      Idist = I
      GOTO 730
    END IF
  CONTINUE

  Nrand = (RAN2(IDUM2) * Numb(idist)) + 1  ! CHOOSE DEME AT THE GIVEN DISTANCE

* LOCATE THE DEME CHOSEN
* ************************************************************
  iX = X0 + DX(idist, nrand)
  iY = Y0 + DY(idist, nrand)
  IF(iX .gt. Xmax) iX = iX - Xmax
  IF(iX .lt. 1) iX = iX + Xmax
  IF(iY .gt. Ymax) iY = iY - Ymax
  IF(iY .lt. 1) iY = iY + Ymax
  IdemeM = iX + (Xmax * (iY - 1))

* The following lines are the Rodeo subroutine (DJ 7-23-12)
  IF(igen.lt.19)THEN
IdemeM=IdemeA !if demic breeding

* IdemeM=RAN2(IDUM2) * 10 + 1 !if panmictic breeding

ELSE IF((igen.eq.19).or.(igen.eq.20))THEN

* IdemeM=11 !for use if everyone moves to one deme

IdemeM=MOD(counter,4) + 11 !for use if 4 demes (11,12,13,14)

* IdemeM=MOD(counter,10) + 11 !for use if 10 demes (11-20)

ELSE

END IF

END

SUBROUTINE MIGRANT STORAGE (Migrant, Adolescent, SexM, rdist,
 + SexA, headcountM, headcountA, idemeA, indivA, idemeM, idist)

* ***************************************************************

INCLUDE + ['drew.fortran.experiment/calldv.FOR'
 + INTEGER Migrant (maxdeme,maxheadroom,maxchromo,maxlocus)
 + INTEGER Adolescent (maxdeme,maxheadroom,maxchromo,maxlocus)
 + INTEGER SEXA (maxdeme, maxheadroom)
 + INTEGER SEXM (maxdeme, maxheadroom)
 + INTEGER Headcounta (maxdeme)
 + INTEGER Headcountm (maxdeme)
 + INTEGER Idemem
 + INTEGER Indivm
 + INTEGER Icopym
 + INTEGER Locusm
 + INTEGER Idemea
 + INTEGER Icopya
 + INTEGER Indiva
 + INTEGER Locusa
 + INTEGER IDIST

REAL RDIST

HeadcountM(idemeM) = HeadcountM(idemeM) + 1

IndivM = headcountM(idemeM)

SexM (idemeM, indivM) = SexA (idemeA, indivA)

DO 820 locusA = 1, maxlocus

locusM = locusA

DO 830 icopyA = 1, maxchromo

icopyM = icopyA

Migrant (idemeM, indivM, icopyM, locusM) =

Adolescent (idemeA, indivA, icopyA, locusA)

CONTINUE

820 CONTINUE

END

SUBROUTINE GAMMA MIGRATION (ProbG)
INCL
REAL ProbG (0:maxXdist)
REAL Q
INTEGER M

DO 920 M = 0, maxXdist
  Q = 1 - P
  IF (M .eq. 0) then
    ProbG(M) = Q**M * P
  ELSE
    ProbG(M) = Q**M * P + ProbG(M-1)
  END IF
920 CONTINUE
END

SUBROUTINE INITIALIZE Y AND A (Young, Adolescent,
HeadcountY, HeadcountA, SexY, SexA)

INCLUDE
* *******************************************************
* The following variables and parameters are the same as in the main program
* *******************************************************
INTEGER Adolescent (maxdeme,maxheadroom,maxchromo,maxlocus)
INTEGER Young (maxdeme,maxheadroom,maxchromo,maxlocus)
INTEGER SEXY (maxdeme, maxheadroom)
INTEGER SEXA (maxdeme, maxheadroom)
INTEGER Headcounty (maxdeme)
INTEGER Headcounta (maxdeme)
INTEGER Idemey
INTEGER Indivy
INTEGER Icopyy
INTEGER locusy
INTEGER Idemea
INTEGER Indiva
INTEGER Icopya
INTEGER Locusa

* *******************************************************
* This Subroutine initializes the genotypes and sexes of the young and adolescent
* arrays see flow chart G
* *******************************************************

* The first thing to do is set the population counters to zero

DO 5 Idemey = 1, maxdeme !loop through all demes
  idemea = idemey
  HeadcountY (idemey) = 0
  headcountA (idemeA) = 0
DO 15 Indivy = 1, maxheadroom !loop through all individuals
  indivA = indivY
  SexA (idemea,indivA) = -1
  SexY (idemey,indivY) = -1 !-1 indicates that the individual is nonexistent
DO 25 icopyY = 1, maxchromo
  icopyA = icopyY
SUBROUTINE MATING (Migrant, Young, SexM, SexY, totind, + HeadcountM, HeadcountY, IDUM, IDUM2, ISEED, igen)

INCLUDE + ' [drew.fortran.experiment] calldv.FOR'

INTEGER totind
INTEGER IdemeM
INTEGER IdemeY
INTEGER INDIVY
INTEGER maxoffspring
INTEGER newind
INTEGER numbmales
INTEGER numbfemales
INTEGER male (maxheadroom, maxchromo, maxlocus)
INTEGER female (maxheadroom, maxchromo, maxlocus)
INTEGER SEXY (maxdeme, maxheadroom)
INTEGER SEXM (maxdeme, maxheadroom)
INTEGER HeadcountY (maxdeme)
INTEGER HeadcountM (maxdeme)
INTEGER Migrant (maxdeme, maxheadroom, maxchromo, maxlocus)
INTEGER Young (maxdeme, maxheadroom, maxchromo, maxlocus)

DO 105 idemeM = 1, maxdeme
idemeY = idemeM
CALL SEX SORTING (numbmales, numbfemales, Male, Female, + SexM, idemem, HeadcountM, Migrant)
IF (numbmales .ne. 0) THEN
IF (numbfemales .ne. 0) THEN
CALL PARENTAL PLANNING (HeadcountM, + Maxoffspring, idemem, igen)
ELSE
maxoffspring = 0
END IF
ELSE
maxoffspring = 0
END IF
DO 115 newind = 1, maxoffspring
CALL MATCHMAKER(idemeM, IDUM, IDUM2, ISEED, + newind, numbfemales, numbmales, Male, + Female, Young, sexY, indivY)
totind = totind + 1
HeadcountY(idemeY) = HeadcountY(idemeY) + 1
115 CONTINUE
105 CONTINUE
END

SUBROUTINE PARENTAL PLANNING (HeadcountM, Maxoffspring,
+ idemem,igen)

INCLUDE
+ [drew.fortran.experiment]calldv.FOR'
INTEGER HeadcountM(maxdeme)
  INTEGER maxoffspring
  INTEGER idemem
REAL Rhold

  rhold = r
*  IF(igen.eq.20) THEN
*    rhold = 2.0
*  END IF

  Maxoffspring = headcountM(idemem) + (rhold * headcountM(idemem) +
  *(maxindiv - headcountM(idemem)) / maxindiv)

  IF(igen.lt.20) THEN
    Maxoffspring = headcountM(idemem) + (rhold * headcountM(idemem) +
    *(10 - headcountM(idemem)) / 10)
  ELSE
    END IF
  IF((igen.eq.20).or.(igen.eq.19)) maxoffspring=100
  IF(igen.lt.19) maxoffspring=10

  IF (maxoffspring >.gt. maxheadroom) maxoffspring = maxheadroom
END

SUBROUTINE SEX SORTING (numbmales, numbfemales, Male, +
  Female, Sexm, idemem, HeadcountM, Migrant)

INCLUDE
+ [drew.fortran.experiment]calldv.FOR'
INTEGER Migrant (maxdeme,maxheadroom,maxchromo,maxlocus)
INTEGER Female (maxheadroom,maxchromo,maxlocus)
INTEGER Male (maxheadroom,maxchromo,maxlocus)
INTEGER SexM (maxdeme, maxheadroom)
INTEGER HeadcountM (maxdeme)
INTEGER numbmales
INTEGER numbmales
INTEGER numbmales
INTEGER headcountM (maxdeme)
INTEGER idemem
INTEGER IndivM
INTEGER locusM
INTEGER ICOPYM

  numbmales = 0
  numbmales = 0

  DO 205 Indivm = 1, headcountM (idemem)
    IF (SexM (idemem, Indivm).eq. 1) THEN
      numbmales = numbmales + 1
    DO 215 locusm = 1, maxlocus
DO 225 icopyM = 1, maxchromo
    Female (numbfemales,icopyM, locusM) = Migrant(idemeM, indivM, icopyM, locusM)
  225      CONTINUE
  215    CONTINUE
ELSE
    numbmales = numbmales + 1
    DO 235 locusm = 1, maxlocus
        DO 245 icopyM = 1, maxchromo
            Male (numbmales,icopyM, locusM) = Migrant(idemeM, indivM, icopyM, locusM)
        245      CONTINUE
    235    CONTINUE
END IF
205    CONTINUE
END

SUBROUTINE MATCHMAKER(idemeM, IDUM,IDUM2,ISEED, newind, + numbmales, numbmales, Male, Female, Young, SexY,INDIVY)
INCLUDE + '[drew.fortran.experiment]calldv.FOR'
INTEGER belle
INTEGER beau
INTEGER IdemeY
INTEGER IndivY
INTEGER IcopyY
INTEGER locusY
    INTEGER IdemeM
    INTEGER IcopyM
INTEGER locusM
    INTEGER newind
INTEGER numbmales
INTEGER numbmales
INTEGER male (maxheadroom, maxchromo, maxlocus)
    INTEGER female (maxheadroom, maxchromo, maxlocus)
INTEGER SexY (maxdeme, maxheadroom)
INTEGER Young(maxdeme,maxheadroom,maxchromo,maxlocus)
belle = Ran2 (IDUM2) * numbmales + 1
beau = Ran2 (IDUM2) * numbmales + 1
idemeY = idemeM
DO 305 icopyY = 1, maxchromo
    indivy = newind
    IF (mod (icopyY,2) .eq. 0) THEN
        DO 315 locusy = 1, maxlocus
            locusM = locusY
            IcopyM = (IRBIT2(ISEED) + 1)
            Young (idemeY, indivY, icopyY, locusY) = female (belle, icopyM, locusM)
            CALL MUTATION (Young, IDUM,IDUM2,ISEED, + idemeY,indivY, icopyY, locusY)
        315      CONTINUE
    ELSE
        DO 325 locusy = 1, maxlocus
            locusM = locusY
    325      CONTINUE
ENDIF
ENDIF
END
icopyM = (IRBIT2(ISEED) + 1)
IF (icopyM.eq.3) icopyM = 2
   Young (idemeY, indivY, icopyY, locusY) =
      Male (beau, icopyM, locusM)
   CALL MUTATION (Young, IDUM,IDUM2,ISEED,
      + idemeY, indivY, icopyY, locusY)
 CONTINUE
325 END IF
305 CONTINUE
SexY (idemeY, indivY) = (IRBIT2(ISEED) + 1)
END

SUBROUTINE MUTATION (Young, IDUM,IDUM2,ISEED,
+ idemeY, indivY, icopyY, locusY)
*
* this subroutine changes the allelic value of the copy at a locus with a frequency of
* 1/mutation rate
* *
INCLUDE + '[drew.fortran.experiment]calldv.FOR'
INTEGER IdemeY
INTEGER IndivY
INTEGER IcopyY
INTEGER locusY
   INTEGER mute
   INTEGER Young(maxdeme,maxheadroom,maxchromo,maxlocus)

   mute = Ran1(IDUM) * mutationrate + 1
IF (mute .eq. mutationrate) THEN
   IF (Young(idemeY, indivY, icopyY, locusY) .eq. 1) THEN
      Young(idemeY, indivY, icopyY, locusY) = 2
   ELSE
      IF (Young(idemeY, indivY, icopyY, locusY) .eq. 2)
         Young(idemeY, indivY, icopyY, locusY) = 1
   ENDIF
ENDIF
END

SUBROUTINE STAT (igen, young, totind, headcountY, iskip, itrial,
+ peakht, idum)

INCLUDE + '[drew.fortran.experiment]calldv.FOR'
INTEGER idemey
INTEGER indivy
INTEGER itrait
INTEGER totind
INTEGER popind
INTEGER IGEN
INTEGER ISKIP
INTEGER IIT
INTEGER IJT
INTEGER LOCUSY
INTEGER ICOPYY
INTEGER I
INTEGER ITRIAL
INTEGER Headcounty(maxdeme)
INTEGER Young(maxdeme,maxheadroom,maxchromo,maxlocus)
INTEGER IndivDATA(0:(2*maxlocus-1))
INTEGER geno(0:(2*maxlocus-1))
REAL fitness
INTEGER hetadv
INTEGER delrec
INTEGER load
INTEGER Hetstart
INTEGER t
   INTEGER Delrecstart
REAL IP2
   INTEGER Iplace
REAL prob
REAL peakht(maxdeme)
   REAL pk
REAL qtotal
   REAL hettotal
   REAL deltotal
   REAL fittotal
   REAL demq(maxdeme)
REAL demhethet(maxdeme)
REAL demdel(maxdeme)
REAL demfit(maxdeme)
REAL popq
REAL pophet
REAL popdel
REAL popfit
REAL popavgq
REAL popavghet
REAL popavgdel
REAL popavgfit
INTEGER qcounter
   REAL M
   REAL B
REAL minfit
REAL maxfit
REAL peakmin
REAL peakmax
REAL popvarq
REAL popvarhet
REAL popvardel
REAL popvarfit
REAL btwvarq
REAL btwvarhet
REAL btwvardel
   REAL btwvarfit
REAL indvarq
REAL indvarhet
   REAL indvardel
   REAL indvarfit
REAL spopvarq
REAL spopvarhet
REAL spopvardel
REAL spopvarfit
REAL sbtwvarq
REAL sbtwvarhet
REAL sbtwvardel
    REAL sbtwvarfit
REAL ssbtwvarq !sums of squares
REAL ssbtwvarhet ! ss
REAL ssbtwvardel ! ss
REAL ssbtwvarfit ! ss
REAL sindvarq
REAL sindvarhet
    REAL sindvardel
    REAL sindvarfit
REAL popavginb
REAL indivfit(maxdeme,maxheadroom)
REAL indivq(maxdeme,maxheadroom)
    REAL indivdel(maxdeme,maxheadroom)
    REAL indivhet(maxdeme,maxheadroom)
INTEGER nodemes
INTEGER countdemes
INTEGER indperdem
INTEGER filenum
INTEGER popnum
CHARACTER*5 breeding
CHARACTER*5 zooseed
INTEGER seperate
REAL btwavgq
REAL btwavghet
REAL btwavgdel
REAL btwavgfit
INTEGER qsum
INTEGER dc

    popnum = mod(itrial,20)
IF(popnum.eq.0) popnum=20

IF(itrial.le.20)THEN
    breeding = 'inbred'
    zooseed = 'deme'
* ELSE IF((itrial.gt.10).and.(itrial.le.20))THEN
*    breeding = 'demic'
*    zooseed = 'demic'
* ELSE IF((itrial.gt.20).and.(itrial.le.30))THEN
*    breeding = 'pan'
*    zooseed = 'rand'
ELSE
    breeding = 'pan'
    zooseed = 'demic'
END IF

minfit=((1*L)-S)
maxfit=(1*L) 
peakmin=0 
peakmax=(2*maxqlocus) 
prob = 0 
popq=0 
pophet=0 
popdel=0 
popfit=0 
sbtwvarq=0 
sbtwvarhet=0 
sbtwvardel=0 
sbtwvarfit=0 
spopvarq=0 
spopvarhet=0 
spopvardel=0 
spopvarfit=0 
nodemes=0 
popind=0 
countdemes=0 

IF ((mod (igen,iskip).eq. 0).or.(igen.le.20).or.(igen.eq.21)) THEN

WRITE(60,2001)itrial,igen 
WRITE(61,2001)itrial,igen 

DO 3533 dc=1,maxdeme 
  IF(headcounty(dc).gt.1)countdemes=countdemes+1 
3533   CONTINUE 

WRITE(60,2002)countdemes 
WRITE(61,2007)countdemes 
WRITE(60,2003) 
WRITE(61,2003) 

DO 3000 idemey=1,maxdeme 
  popind=popind+headcountY(idemey) 
  pk= peakht(idemey) 
  qtotal=0 
  hettotal=0 
  delttotal=0 
  fittotal=0 
  IF(headcountY(idemey).gt.1)THEN 
    WRITE(60,2004)idemey 
    WRITE(61,2004)idemey 
  ELSE 
    END IF 
3000   CONTINUE 

ELSE 
  END IF 

DO 3200 indivy=1,headcounty(idemey) 
  indivq(idemey,indivy)=0 
  qsum=0 
  DO 3250 j=1,(2*maxlocus) 
    IndivDATA(j)=0 
3250     CONTINUE 

Iplace = 0
ib=-1

DO 3309 locusy=1,maxlocus
  DO 3310 icopyY=1,maxchromo
    ib=ib+1
    geno(ib)=young(idemeY,indivY,icopyY,locusY)
    CONTINUE
  CONTINUE

DO 3300 locusy=1,maxqlocus !find the genotype
  DO 3400 icopyY=1,maxchromo
    Iplace= + Iplace+Young(idemeY,indivY,icopyY,locusY)-1
  CONTINUE

IF(young(idemeY,indivY,1,locusY).ne.young(idemeY, + indivY,2,locusY))THEN
  qsum=qsum+1
ELSE
  qsum=qsum
END IF

indivq(idemey,indivy)=qsum

IF(headcountY(idemeY).gt.1)THEN
  WRITE(60,2010)indivy,(geno(j2),j2=0,31)
  WRITE(61,2011)indivy,(geno(j1),j1=32,231)
ELSE
  END IF

IP2=Iplace

IF (Ip2 .lt. pk)THEN
  M = ((maxfit-minfit)/(peakht(idemeY)-peakmin))
  prob= ((M*IP2)+minfit)
ELSE IF (IP2.eq.peak)THEN
  prob= maxfit
ELSE
  M=((minfit-maxfit)/(peakmax-peakht(idemeY)))
  prob= ((M*(IP2-peakht(idemeY)))+maxfit)
END IF

hetadv = 0
delrec = 0
Hetstart= (maxqlocus+1)
Delrecstart= (maxhetadv +1)

DO 3100 locount = Hetstart, maxhetadv !counts the impact of all het adv loci
  IF (YOUNG(idemey,indivy,1,locount) .eq. + YOUNG(idemey,indivy,2,locount)) THEN
hetadv = hetadv + 1

ELSE
ENDIF

CONTINUE

DO 3150 hicount = Delrecstart,maxlocus
IF ((YOUNG(idemey, indivy, 1, hicount).eq. 1) +
and. (YOUNG(idemey, indivy, 2, hicount).eq. 1)) THEN
delrec = delrec + 1

END IF

CONTINUE

fitness = (prob-(inbeffect*(delrec + hetadv))/maxfit)
indivfit(idemey,indivy)=fitness
indivhet(idemey,indivy)=hetadv
indivdel(idemey,indivy)=delrec
popfit=popfit+fitness
popdel=popdel+delrec
pophet=pophet+hetadv

ia=-1
DO 2017 ilocus=1,maxlocus
DO 2018 icopy=1,maxchromo
ia=ia+1
IndivDATA(ia)= YOUNG(idemey,indivy,icopy,Ilocus)

CONTINUE

CONTINUE

max2=2*maxlocus-1

qtotal=qtotal+(indivq(idemey,indivy)/16)
hettotal=hettotal+hetadv
deltotal=deltotal+delrec
fittotal=fittotal+fitness

CONTINUE

IF(headcountY(idemey).gt.0)THEN
demq(idemey)=(qtotal/headcountY(idemey))
demhet(idemey)=(hettotal/headcountY(idemey))
demdel(idemey)=(deltotal/headcountY(idemey))
demfit(idemey)=(fittotal/headcountY(idemey))
popq=popq+qtotal
ELSE
ENDIF

WRITE(60,2013)
WRITE(61,2013)

CONTINUE
WRITE(60,2005)
WRITE(60,2006)
WRITE(61,2005)
WRITE(61,2006)

IF(popind.ne.0)THEN
popavqg=(popq/popind)
popavghet=(pophet/popind)
popavgdel=(popdel/popind)
popavgfit=(popfit/popind)
popavginb=((popavghet+popavgdel)*inbeffect) !the average inbreeding

DO 7701 idemey=1,maxdeme
   sindvarq=0
   sindvarhet=0
   sindvardel=0
   sindvarfit=0

DO 7702 indivy=1,headcountY(idemey)
   IF(headcountY(idemey).gt.0)THEN
      spopvarq=spopvarq+(((indivq(idemey,indivy)/16) - popavqg)**2)
      spopvarhet=spopvarhet+((indivhet(idemey,indivy) - popavghet)**2)
      spopvardel=spopvardel+((indivdel(idemey,indivy) - popavgdel)**2)
      spopvarfit=spopvarfit+((indivfit(idemey,indivy) - popavgfit)**2)
      sindvarq=(sindvarq+((demq(idemey)-
                      (indivq(idemey,indivy)/16))**2))
      sindvarhet=(sindvarhet+(demhet(idemey)-
                      (indivhet(idemey,indivy))**2))
      sindvardel=(sindvardel+(demdel(idemey)-
                      (indivdel(idemey,indivy))**2))
      sindvarfit=(sindvarfit+(demfit(idemey)-
                      (indivfit(idemey,indivy))**2))
   ELSE
      END IF
   CONTINUE

7702  CONTINUE

   IF(headcountY(idemey).GT.0)THEN
      nodemes=nodemes+1
      sbtwvarq=(sbtwvarq+demq(idemey))
      sbtwvarhet=(sbtwvarhet+demhet(idemey))
      sbtwvardel=(sbtwvardel+demdel(idemey))
      sbtwvarfit=(sbtwvarfit+demfit(idemey))
      indvarq=(sindvarq/headcountY(idemey)) !variance for quantitative trait for whole deme
      indvarhet=(sindvarhet/headcountY(idemey)) !and for # heterozygotically disadvantageous loci
\[ \text{indvardel} = (\text{sindvardel/headcountY(idemeY)}) \quad \text{!and # deleterious recessive loci} \\
\text{indvarfit} = (\text{sindvarfit/headcountY(idemeY)}) \quad \text{!and for fitness} \]

Write \((10,5005)\) trial, Igen, idemeY, headcountY(idemeY), 
+ demfit(idemeY), indvarfit, demhet(idemeY), indvarhet, 
+ demdel(idemeY), indvardel, demq(idemeY), indvarq, 
+ peakht(idemeY), idum, popnum, 4, breeding, zooseed, 
+ programe, qsurf

ELSE
END IF

7701 CONTINUE

\[
\begin{align*}
\text{btwavgq} &= (\text{sbtwvarq/nodemes}) \\
\text{btwavghet} &= (\text{sbtwvarhet/nodemes}) \\
\text{btwavgdel} &= (\text{sbtwvardel/nodemes}) \\
\text{btwavgfit} &= (\text{sbtwvarfit/nodemes}) \\
\text{ssbtwvarq} &= 0 \\
\text{ssbtwvarhet} &= 0 \\
\text{ssbtwvardel} &= 0 \\
\text{ssbtwvarfit} &= 0 \\
\end{align*}
\]

DO 7705 idemeY=1, maxdeme 
IF(headcountY(idemeY).gt.0) THEN 
\[
\begin{align*}
\text{ssbtwvarq} &= \text{ssbtwvarq} + ((\text{btwavgq - demq(idemeY)})**2) \\
\text{ssbtwvardel} &= \text{ssbtwvardel} + ((\text{btwavgdel - demdel(idemeY)})**2) \\
\text{ssbtwvarhet} &= \text{ssbtwvarhet} + ((\text{btwavghet - demhet(idemeY)})**2) \\
\text{ssbtwvarfit} &= \text{ssbtwvarfit} + ((\text{btwavgfit - demfit(idemeY)})**2) \\
\end{align*}
\]
ELSE 
END IF 

7705 CONTINUE

\[
\begin{align*}
\text{btwvarq} &= (\text{ssbtwvarq/nodemes}) \\
\text{btwvarhet} &= (\text{ssbtwvarhet/nodemes}) \\
\text{btwvardel} &= (\text{ssbtwvardel/nodemes}) \\
\text{btwvarfit} &= (\text{ssbtwvarfit/nodemes}) \\
\end{align*}
\]

\[
\begin{align*}
\text{popvarq} &= (\text{spopvarq/popind}) \quad \text{!variance for quantitative trait for whole population} \\
\text{popvarhet} &= (\text{spopvarhet/popind}) \quad \text{!and for # heterozygotically disadvantageous loci} \\
\text{popvardel} &= (\text{spopvardel/popind}) \quad \text{!and # deleterious recessive loci} \\
\text{popvarfit} &= (\text{spopvarfit/popind}) \quad \text{!and for fitness} \\
\end{align*}
\]
Write (9,5000) itrial,igen,popind,nodemes,popavgfit,popvarfit,
+ btwvarfit,popavq,popvarq,btwvarq,popavghet,popvarhet,
+ btwvarhet,popavdel,popvardel,btwvardel,popavinb,
+ idum,popnum,4,breeding,zooseed,progname,qsurf

ELSE
END IF

ELSE
END IF

WRITE(60,2005)
WRITE(61,2005)

5000 FORMAT (1x,16,3(',','1x,i6),13(',','2x,f10.5),',',i8, + ',',i2,','','i1,','','A5,','','A5',',','A10',',','A2)

5005 FORMAT (1x,i5,1x,(3(',','I5,1x)),9(',','f10.5,1x), + 3(',','I8,1x),A6,','','A6',','','A10',','','A2)

2001 FORMAT([''pop #='','i3,2x,'generation =','i3,''])

2002 FORMAT(6x,'dimensions',1x,'nloci=16',1x,'npops=','i3,','')

2007 FORMAT(6x,'dimensions',1x,'nloci=100',1x,'npops=','i3,','')

2003 FORMAT('matrix')

2010 FORMAT('''indiv_''',i3,''''',2x,32(i2))

2004 FORMAT('''pop''',i3,''''','')

2011 FORMAT('''indiv_''',i3,''''',2x,100(i2))

2005 FORMAT('';)

2006 FORMAT(3X,'end;')

2013 FORMAT(';')

END

FUNCTION RAN1(IDUM)
* *******************************************************
* RETURNS A UNIFORM RANDOM DEViate BETWEEN 0.0 AND 1.0 SEt IDUM TO ANY
* NEGATIVE VALUE TO INITIALIZE OR REINITIALIZE THE SEQUENCE.
* THIS ROUTINE IS TAKEN FROM NUMERICAL RECIPIES (FORTRAN) BY PRESS
* FLANNERY, TEUKOLSKY AND VETTERLING (CAMBRIDGE PRESS). THIS IS
* SLOWER THAN RAN2
* *******************************************************
REAL*8 R(97)
PARAMETER (M1=134456,IA1=8121, IC1=28411, RM1=1.0/M1)
PARAMETER (M2=243000,IA2=4561, IC2=51349, RM2=1.0/M2)
PARAMETER (M3=259200,IA3=7141, IC3=54773)
DATA IFF/0/

IF (IDUM .LT. 0 .OR. IFF .EQ. 0)THEN
  IFF = 1
  IX1=MOD(IC1-IDUM,M1) ! SEED THE FIRST ROUND
  IX1=MOD(IA1*IX1+IC1,M1)
  IX2=MOD(IX1,M2) ! USE IT TO SEED THE SECOND
  IX1=MOD(IA1*IX1+IC1,M1)
  IX3=MOD(IX1,M3) ! AND THE THIRD
  DO 11 J = 1, 97 ! FILL THE TABLE WITH UNIFORM DEVIATES
  IX1=MOD(IA1*IX1+IC1,M1) ! provided by first two routines
  IX2=MOD(IA2*IX2+IC2,M2)
  R(J)= (FLOAT(IX1)+FLOAT(IX2)*RM2)*RM1 !Combine the low and high order peices here
  CONTINUE
  IDUM = 1
ENDIF
IX1=MOD(IA1*IX1+IC1,M1) ! BEGIN MORMAL DRAWS
IX2=MOD(IA2*IX2+IC2,M2)
IX3=MOD(IA3*IX3+IC3,M3)
J=1+(97*IX3)/M3
IF(J .GT. 97 .OR. J .LT. 1) PAUSE
RAN1=R(J)
R(J)= (FLOAT(IX1)+FLOAT(IX2)*RM2)*RM1
RETURN
END

FUNCTION RAN2(IDUM2)
  * ************************************************************
  * RETURNS A UNIFORM RANDOM DEVIATE BETWEEN 0.0 AND 1.0 SET IDUM2 TO ANY
  * NEGATIVE VALUE TO INITIALIZE OR REINITIALIZE THE SEQUENCE.
  * THIS ROUTINE IS TAKEN FROM NUMERICAL RECIPIES (FORTRAN) BY PRESS
  * FLANNERY, TEUKOLSKY AND VETTERLING (CAMBRIDGE PRESS). THIS IS
  * FASTER THAN RAN1 BUT IT GENERATES LESS RANDOM NUMBERS SO DO NOT RELY
  * ON IT FOR NUMBERS THAT REQUIRE NEARLY ALL THE SIGNIFICANT BITS OF WORD
  * SIZE
  * AVAILABLE. SPECIFICALLY. I WOULD USE CAUTION BEYOND THE FOURTH
  * SIGN.DIG.
  * ************************************************************

INTEGER IR(97)
PARAMETER (M=714025,IA=1366, IC=1508, RM=1.0/M)
DATA IFF/0/ ! initialize on the first round

IF (IDUM2 .LT. 0 .OR. IFF .EQ. 0)THEN
  IFF = 1
  IDUM2=MOD(IC-IDUM2,M) ! SEED THE FIRST ROUND
  DO 11 J = 1, 97 ! FILL THE TABLE WITH UNIFORM DEVIATES
  IDUM2=MOD(IA*IDUM2+IC,M)
  IR(J)= IDUM2
CONTINUE
IDUM2=MOD(IA*IDUM2+IC,M)
  IY = IDUM2
ENDIF

J=1+(97*IY)/M  ! BEGIN HERE EXCEPT ON INITIALIZATION
IF(J.GT.97 .OR. J.LT.1) PAUSE
  IY=IR(J)
RAN2=IY*RM
IDUM2=MOD(IA*IDUM2+IC,M)
IR(J)=IDUM2
RETURN
END

FUNCTION IRBIT2(ISEED)
  * **************************************************************
  * RETURNS A RANDOM BIT AS AN INTEGER (1, OR 0), BASED ON THE 30 LOW
  * SIGNIFICANCE BITS IN ISEED. ISEED SHOULD BE SET TO A LARGE NEGATIVE
  * INTEGER TO INITIALIZE OR REINITIALIZE THE SEQUENCE.
  * THIS ROUTINE IS TAKEN FROM NUMERICAL RECIPIES (FORTRAN) BY PRESS,
  * FLANNERY, TEUKOLSKY AND VETTERLING (CAMBRIDGE PRESS).
  * NOTE THAT THIS IS FUNCTION IS NOT WRITEN IN FORTRAN 77, IT USES THE
  * VAX-11 FUNCTIONS IXOR (BITWISE EXCLUSIVE OR) ISHFT (BITWISE SHIFT) AND
  * IAND (BITWISE AND) I WOULD EXPECT THAT THESE ARE PRETTY
  * COMMON EXTENTIONS HOWEVER.
  * **************************************************************

  PARAMETER(ib1=1,ib4=8,ib6=32,ib30=536870912,mask=ib1+ib4+ib6)
  IF (IAND(iseed, ib30) .NE. 0) THEN  ! change all the masked bits, shift, and put 1 into bit 1.
    iseed=IOR(ISHFT(IEOR(iseed,mask), 1),IB1)
    irbit2=1
  ELSE
    iseed=IAND(ISHFT(iseed,1),NOT(IB1))
    irbit2=0
  ENDIF
  RETURN
END
APPENDIX C

PARAMETER FILE FOR SEED POPULATIONS

The following is the parameter file that contained many of the global variables used in the simulation during creation of the seed populations.

REAL P
REAL K
REAL S
REAL L
REAL r
REAL HT
REAL LT
REAL LPL
REAL LPH
REAL HPL
REAL HPH
REAL RAN1
REAL RAN2
real tz
real sz
REAL petinbred  !what % of initial inbreeding loci will be dedicated to heterozygote advantage
   INTEGER IRBIT2
INTEGER MAXT
INTEGER  MAXGEN
INTEGER MAXDEME
INTEGER YMAX
INTEGER XMAX
INTEGER MAXXDIST
INTEGER MAXYDIST
INTEGER MAXNUMB
INTEGER MAXINDIV
INTEGER HEADROOM
INTEGER MAXHEADROOM
INTEGER MAXALLELE
INTEGER MAXLOCUS  !The total number of all loci
   INTEGER MAXQLOCUS  !the number of quantitative trait loci
   REAL INBeffect
INTEGER MAXCHROMO
INTEGER MUTATIONRATE
INTEGER IDUM
INTEGER IDUM2
INTEGER ISEED
REAL peak
   INTEGER Dominance
INTEGER maxinbred  !The number of inbreeding loci
INTEGER maxhetadv  !counter for het advantage
INTEGER release

INTEGER maxzoodeme
INTEGER maxzoogen
INTEGER maxzoohheadroom
INTEGER sel
CHARACTER*10 proname
CHARACTER*2 qsurf

* Maxgen determines how many generations the simulation will run for.
PARAMETER (maxgen = 5000)
PARAMETER (maxzoogen = 19)

* Dominance indicates whether full dominance (1) or no dominance (0) exists for the delterious loci
PARAMETER (dominance = 0)
PARAMETER (maxzoodeeme = 10)
PARAMETER (maxzoohheadroom = 10)
PARAMETER (release = 1)
PARAMETER (sel = 1) !sel = 1 for stable envir...0 for variable environment see swright
PARAMETER (proname = 'wild1s.for')
PARAMETER (qsurf = 's')

*********************************************************
* Maxdeme is the number of demes included in the simulation
* if maxdeme is changed xmax and ymax must also be changed.
PARAMETER (maxdeme = 100)
* Ymax & Xmax are the maximum number of demes in the x and y directions.
* Xmax * Ymax must therefore equal maxdeme (X must be > or = Y also).
PARAMETER (ymax = 10)
PARAMETER (xmax = 10)

* Individuals can only migrate so far in the Y or X direction.
PARAMETER (maxXdist = Xmax/2)
PARAMETER (MaxYdist = Ymax/2)

* Maxnumb = the most demes which might be found at any distance
PARAMETER (maxnumb = Xmax*Xmax)

*********************************************************
* Maxindiv = the maximum number of individuals in a deme
PARAMETER (maxindiv = 100)
* headroom = the maximum number of individuals over the carrying capacity that
* may temporarily exist
PARAMETER (headroom = maxindiv/2)
* maxheadroom = the total number of individuals a population can temporarily
* support
PARAMETER (maxheadroom = maxindiv + headroom)

*********************************************************
* The following parameters should be adjusted to reflect the number of loci,
* chromosomes, and alleles at a loci which are possible.
PARAMETER (maxallele = 2)
PARAMETER (maxqlocus = 16)
PARAMETER (maxchromo = 2)

PARAMETER (maxinbred = 100) ! number of inbreeding loci

PARAMETER (inbeffect = .02) ! impact on fitness of a del recessive loci usuall .02

PARAMETER (pctinbred = 0) ! percentage of inbreeding loci attributed to het adv/del recessives

PARAMETER (maxhetadv = (pctinbred*maxinbred)+maxqlocus)

PARAMETER (maxlocus = maxqlocus + maxinbred)

PARAMETER (maxt = maxchromo*maxqlocus)

* The mutation rate (mutationrate) is the frequency of mutation per copy
* of a locus.

PARAMETER (mutationrate = 100000)

******************************************************************************

** The following parameters are representing the components of viability selection. S**
* represents the selective disadvantage least advantageous genotype, and when multiplied
* by K gives the advantage of most favored genotype. L is the mortality which is in
* independent of the genes being looked at (L will effect drift rates).

******************************************************************************

PARAMETER (R=1.1)
PARAMETER (K=1)
PARAMETER (S=0.4)
PARAMETER (L=.9)
PARAMETER (tz=16)
PARAMETER (sz=16)
PARAMETER (LT=0)
PARAMETER (HT=0)
PARAMETER (HPH=16)
PARAMETER (HPL=16)
PARAMETER (LPL=16)
PARAMETER (LPH=16)
PARAMETER (peak=16.0) ! this variable will be used to define location of single peak

* 1+KS + - - - - - - ****
* |   * |   *
* 1+KS-LT + - * |   *
* |   |   *
* 1 + - - - | - - | - - - - - - ****
* |   |   * |   *
* 1-HT + - - - | - - * - - | - - *
* |   |   * * |   *
* 1-S + - - | - - | - - **** |   *
* +------++------++------++------++------
* 0 H 1 H S T L L
* P P Z Z P P
* L H L H

************************************************************************************

* P is 1 - the probability of an individual migrating distance 1

************************************************************************************

PARAMETER (P = .9995)
APPENDIX D
PARAMETER FILE FOR CAPTIVE RUNS

The following is the parameter file that contained many of the global variables used in the simulation during captivity and release into the wild. These particular variables were set for a structured captive populations and a variable release environment.

REAL P
REAL K
REAL S
REAL L
REAL r
REAL HT
REAL LT
REAL LPL
REAL LPH
REAL HPL
REAL HPH
REAL RAN1
REAL RAN2
REAL tz
REAL sz
REAL pctinbred !what % of initial inbreeding loci will be dedicated to heterozygote advantage
INTEGER IRBIT2
INTEGER MAXT
INTEGER MAXGEN
INTEGER MAXDEME
INTEGER YMAX
INTEGER XMAX
INTEGER MAXXDIST
INTEGER MAXYDIST
INTEGER MAXNUMB
INTEGER MAXINDIV
INTEGER HEADROOM
INTEGER MAXHEADROOM
INTEGER MAXALLELE
INTEGER MAXLOCUS  !The total number of all loci
INTEGER MAXQLOCUS  !the number of quantitative trait loci
REAL INBeffect
INTEGER MAXCHROMO
INTEGER MUTATIONRATE

INTEGER IDUM
INTEGER IDUM2
INTEGER ISEED
REAL peak
INTEGER Dominance
INTEGER maxinbred  !The number of inbreeding loci
INTEGER maxhetadv  !counter for het advantage
INTEGER release
INTEGER maxzoodeme
INTEGER maxzoogen
INTEGER maxzoheadroom
INTEGER sel
CHARACTER*10 prograde
CHARACTER*2 qsurf

* Maxgen determines how many generations the simulation will run for.
PARAMETER (maxgen = 100)
PARAMETER (maxzoogen = 19)

* Dominance indicates whether full dominance (1) or no dominance (0) exists for the delterious loci
PARAMETER (dominance = 0)
PARAMETER (maxzotheadroom = 100)
PARAMETER (release = 1)
PARAMETER (sel = 1) !sel = 1 for stable envir...0 for variable environment see swright
PARAMETER (prograde = 'wild1s.for')
PARAMETER (qsurf = 's')

************************************************************************************

* Maxdeme is the number of demes included in the simulation
* if maxdeme is changed xmax and ymax must also be changed.
PARAMETER (maxdeme = 100)

* Ymax & Xmax are the maximum number of demes in the x and y directions.
* Xmax * Ymax must therefore equal maxdeme (X must be > or = Y also).
PARAMETER (ymax = 10)
PARAMETER (xmax = 10)

* Individuals can only migrate so far in the Y or X direction.
PARAMETER (maxXdist = Xmax/2)
PARAMETER (MaxYdist = Ymax/2)

* Maxnumb = the most demes which might be found at any distance
PARAMETER (maxnumb = Xmax*Xmax)

**************************************************************************************

* Maxindiv = the maximum number of individuals in a deme
PARAMETER (maxindiv = 100)

* headroom = the maximum number of individuals over the carrying capacity that
* may temporarily exist
PARAMETER (headroom = maxindiv/2)

* maxheadroom = the total number of individuals a population can temporarily
* support
PARAMETER (maxheadroom = maxindiv + headroom)

**************************************************************************************

* The following parameters should be adjusted to reflect the number of loci,
* chromosomes, and alleles at a loci which are possible.
PARAMETER (maxallele = 2)
PARAMETER (maxqlocus = 16)
PARAMETER (maxchromo = 2)
PARAMETER (maxinbred = 100)  !number of inbreeding loci
PARAMETER (inbeffect = .02)  !impact on fitness of a del recessive loci usuall .02

PARAMETER (pctinbred = 0)  !percentage of inbreeding loci attributed to het adv
PARAMETER (maxhetadv = (pctinbred*maxinbred)+maxqlocus))
PARAMETER (maxlocus = maxqlocus + maxinbred)
PARAMETER (maxt = maxchromo*maxqlocus)
* The mutation rate (mutationrate) is the frequency of mutation per copy
* of a locus.
PARAMETER (mutationrate = 100000)
* **********************************************

* The following parameters are representing the components of viability selection. S
* represents the selective disadvantage least advantageous genotype, and when multiplied
* by K gives the advantage of most favored genotype. L is the mortality which is in
* independent of the genes being looked at (L will effect drift rates).
* **********************************************
PARAMETER (R=1.1)
PARAMETER (K=1)
PARAMETER (S=0.4)
PARAMETER (L=.9)
PARAMETER (tz=16)
PARAMETER (sz=16)
PARAMETER (LT=0)
PARAMETER (HT=0)
PARAMETER (HPH=16)
PARAMETER (HPL=16)
PARAMETER (LPL=16)
PARAMETER (LPH=16)
PARAMETER (peak=16.0)  !this variable will be used to define location of single peak

P is 1 - the probability of an individual migrating distance 1
**********************************************
PARAMETER (P = .95)