```
#  Exponential Growth State Space model version date 120815.R:
#
#  Program for calculating maximum likelihood (ML) or restricted maximum
#  likelihood (REML) estimates of unknown parameters for the Exponential
#  Growth State Space (EGSS) model of stochastic population growth.
#  The model is
#
#  X(t) = mu + E(t)
#               with E(t) ~ normal(0,ssq),
#  Y(t) = X(t) + F(t)
#               with F(t) ~ normal(0,tsq).
#
#  Here X(t) is log-population abundance, Y(t) is observed or estimated value
#  of X(t), x0, mu, ssq, tsq are parameters.  The parameter ssq is the
#  variance of the process noise, and tsq is the variance of the observation
#  error.
#  The model takes population abundance N(t) = exp(X(t)) to be governed by a
#  stochastic, density independent model, with the observed
#  abundances O(t) = N(t)*exp(F(t)) arising from lognormal sampling error.
#
#  User provides time series of observed population abundances o(0), o(1),
#  ..., o(q), which are log-transformed by the program into y(0), y(1), ...,
#  y(q), assumed to be a time series realization of Y(t).  Likelihood
#  function of y(0), y(1), ..., y(q) is that of a multivariate normal
#  distribution.  The observation times t_0, t_1, t_2, ..., t_q can have
#  unequal intervals.
#
#  Program computes initial parameter values for iterations.  The program
#  should be re-run for several sets of initial values, as the likelihood
#  function for the model frequently has multiple local maxima.
#
#  An alternative program, in SAS, is available as an online appendix to
#  Staples et al. (2004).
#
#  This program written by Brian Dennis (Dept Fish and Wildlife Sciences,
#  Univ Idaho, Moscow, Idaho, 83844-1136 USA).
#
#  Citations:
#    Staples et al. 2004.  Ecology.
#    Dennis et al. 2006.  Ecological Monographs.
#    Humbert et al.  2009.  Oikos.
#
#-----------------------------------------------------------------------
#        USER INPUT SECTION
#-----------------------------------------------------------------------
#  User supplies time series data here.  User can substitute R statements to
#    read population abundance data from a file into the vector "Observed.t".
#    Times of observation are entered into the vector "Time.t".
# Observed.t=c(18,10,9,14,17,14,5,10,9,5,11,11,4,5,4,8,2,3,9,2,4,7,4,1,2,
#    4,11,11,9,6)  #  No zeros!  (With zeros, you must use another model)
# Time.t=c(0,1,2,3,4,5,6,7,8,9,10,11,12,13,14,15,16,17,18,19,20,21,22,23,24,
#    25,26,27,28,29)  #  Initial time can be nonzero.
#
#  Example data are American Redstart counts from North American Breeding
#    Bird Survey, record # 02014 3328 08636, 1966-95 (Table 1 in Dennis
#    et al. 2006).
```

```
# Polar bear, from Canada
Observed.t=c(1213,1187,1295,1067,1038,969,1233,1027,1236,1039,1055,
  970,1145,875,1144,968,965,869)  #  No zeros!
Time.t=c(1987,1988,1989,1990,1991,1992,1993,1994,1995,1996,1997,1998,
  1999,2000,2001,2002,2003,2004)

#-----------------------------------------------------------------------
#        PROGRAM INITIALIZATION SECTION
#-----------------------------------------------------------------------
library(MASS)   #  loads miscellaneous functions (ginv, etc.)
T.t=Time.t-Time.t[1]  #  Time starts at zero.
Y.t=log(Observed.t)   #  Log-transform the observations.
q=length(Y.t)-1       #  Number of time series transitions, q.
qp1=q+1               #  q+1 gets used a lot, too.
S.t=T.t[2:qp1]-T.t[1:q]   #  Time intervals.
m=rep(1,qp1)   #  Will contain Kalman means for Kalman calculations.
v=rep(1,qp1)   #  Will contain variances for Kalman calculations.

#-----------------------------------------------------------------------
#        SECTION FOR DEFINING ML & REML LOG-LIKELIHOODS
#-----------------------------------------------------------------------

#  ML objective function "negloglike.ml" is negative of log-likelihood;
#  the Nelder-Mead optimization routine in R, "optim", is a minimization
#  routine.  The ML objective function uses equations 24-26 from Dennis et
#  al. (2006).  The three function arguments are:  theta, vector of
parameters
#  (transformed to the real line), yt, vector of time series observations,
and
#  tt, vector of observation times.
negloglike.ml=function(theta,yt,tt) {
   muu=theta[1]
   sigmasq=exp(theta[2])       #  Constrains ssq > 0.
   tausq=exp(theta[3])         #  Constrains tsq > 0.
   xzero=theta[4]
   q=length(yt)-1
   qp1=q+1
   vx=matrix(0,qp1,qp1)
   for (ti in 1:q) {
      vx[(ti+1),(ti+1):qp1]=matrix(1,1,(qp1-ti))*tt[ti+1]
      vx[(ti+1):qp1,(ti+1)]=t(vx[(ti+1),(ti+1):qp1])
   }
   Sigma.mat=sigmasq*vx
   Itausq=matrix(rep(0,(qp1*qp1)),nrow=qp1,ncol=qp1)
   diag(Itausq)=rep(tausq,qp1)
   V=Sigma.mat+Itausq
   mu=xzero+muu*tt
   ofn=((qp1)/2)*log(2*pi)+(0.5*log(det(V)))+
      (0.5*((yt-mu)%*%ginv(V)%*%(yt-mu)))
   return(ofn)
}

#  REML objective function "negloglike.reml" is negative of log-likelihood
#  for second differences of the log-scale observations.  The REML objective
#  function uses equations A22-A25 from Humbert et al. (2009).  The three
#  function arguments are:  theta, vector of parameters (transformed to the
#  real line), yt, vector of time series observations (log scale), and
```

```
#  tt, vector of observation times.  Function performs the differencing.
negloglike.reml=function(theta,yt,tt) {
   sigsq=exp(theta[1])          #  Constrains ssq > 0.
   tausq=exp(theta[2])          #  Constrains tsq > 0.
   q=length(yt)-1
   qp1=q+1
 vx=matrix(0,qp1,qp1)
   for (ti in 1:q) {
      vx[(ti+1),(ti+1):qp1]=matrix(1,1,(qp1-ti))*tt[ti+1]
      vx[(ti+1):qp1,(ti+1)]=t(vx[(ti+1),(ti+1):qp1])
   }
   Sigma.mat=sigsq*vx
   Itausq=matrix(rep(0,(qp1*qp1)),nrow=qp1,ncol=qp1)
   diag(Itausq)=rep(tausq,qp1)
   V=Sigma.mat+Itausq
   ss=tt[2:qp1]-tt[1:q]
   D1mat=cbind(-diag(1/ss),matrix(0,q,1))+cbind(matrix(0,q,1),diag(1/ss))
   D2mat=cbind(-diag(1,q-1),matrix(0,q-1,1))+
      cbind(matrix(0,q-1,1),diag(1,q-1))
   Phi.mat=D2mat%*%D1mat%*%V%*%t(D1mat)%*%t(D2mat)
   wt=(yt[2:qp1]-yt[1:q])/ss
   ut=wt[2:q]-wt[1:q-1]
   ofn=(q/2)*log(2*pi)+(0.5*log(det(Phi.mat)))+
      (0.5*(ut%*%ginv(Phi.mat)%*%ut))
   return(ofn)
}

#-----------------------------------------------------------------------
#        SECTION FOR CALCULATING EGOE AND EGPN ESTIMATES
#        (FOR USE AS INITIAL VALUES)
#-----------------------------------------------------------------------
Ybar=mean(Y.t)
Tbar=mean(T.t)
mu.egoe=sum((T.t-Tbar)*(Y.t-Ybar))/sum((T.t-Tbar)*(T.t-Tbar))
x0.egoe=Ybar-mu.egoe*Tbar
ssq.egoe=0
Yhat.egoe=x0.egoe+mu.egoe*T.t
tsq.egoe=sum((Y.t-Yhat.egoe)*(Y.t-Yhat.egoe))/(q-1)

Ttr=sqrt(S.t)
Ytr=(Y.t[2:qp1]-Y.t[1:q])/Ttr
mu.egpn=sum(Ttr*Ytr)/sum(Ttr*Ttr)
Ytrhat=mu.egpn*Ttr
ssq.egpn=sum((Ytr-Ytrhat)*(Ytr-Ytrhat))/(q-1)
tsq.egpn=0
x0.egpn=Y.t[1]

mu0=(mu.egoe+mu.egpn)/2     #  Initial values for EGSS are averages
ssq0=ssq.egpn/2             #    of EGOE and EGPN values
tsq0=tsq.egoe/2             #            --
x00=x0.egoe                 #            --

#-----------------------------------------------------------------------
#        SECTION FOR CALCULATING ML & REML PARAMETER ESTIMATES
#-----------------------------------------------------------------------

# The ML estimates.
```

```
EGSSml=optim(par=c(mu0,log(ssq0),log(tsq0),x00),
   negloglike.ml,NULL,method="Nelder-Mead",yt=Y.t,tt=T.t)
params.ml=c(EGSSml$par[1],exp(EGSSml$par[2]),exp(EGSSml$par[3]),
   EGSSml$par[4])
lnlike.ml=-EGSSml$value[1]
AIC.egss=-2*lnlike.ml+2*length(params.ml)

mu.ml=params.ml[1]            # These are the ML estimates.
ssq.ml=params.ml[2]           #          --
tsq.ml=params.ml[3]           #          --
x0.ml=params.ml[4]            #          --

# The REML estimates.
EGSSreml=optim(par=c(log(ssq0),log(tsq0)),
   negloglike.reml,NULL,method="Nelder-Mead",yt=Y.t,tt=T.t)
params.reml=c(exp(EGSSreml$par[1]),exp(EGSSreml$par[2]))
ssq.reml=params.reml[1]  #  These are the REML estimates.
tsq.reml=params.reml[2]  #           --



vx=matrix(0,qp1,qp1)
for (ti in 1:q) {
  vx[(ti+1),(ti+1):qp1]=matrix(1,1,(qp1-ti))*T.t[ti+1]
  vx[(ti+1):qp1,(ti+1)]=t(vx[(ti+1),(ti+1):qp1])
}

Sigma.mat=ssq.reml*vx
Itausq=matrix(rep(0,(qp1*qp1)),nrow=qp1,ncol=qp1)
diag(Itausq)=rep(tsq.reml,qp1)
V=Sigma.mat+Itausq
D1mat=cbind(-diag(1/S.t),matrix(0,q,1))+cbind(matrix(0,q,1),diag(1/S.t))
V1mat=D1mat%*%V%*%t(D1mat)
W.t=(Y.t[2:qp1]-Y.t[1:q])/S.t
j1=matrix(1,q,1)
V1inv=ginv(V1mat)
mu.reml=(t(j1)%*%V1inv%*%W.t)/(t(j1)%*%V1inv%*%j1)
j=matrix(1,qp1,1)
Vinv=ginv(V)
x0.reml=(t(j)%*%Vinv%*%(Y.t-mu.reml*T.t))/(t(j)%*%Vinv%*%j)
Var_mu.reml=1/(t(j1)%*%V1inv%*%j1)         #  Variance of mu
mu_hi.reml=mu.reml+1.96*sqrt(Var_mu.reml)  #  95% CI for mu
mu_lo.reml=mu.reml-1.96*sqrt(Var_mu.reml)  #       --

#  Calculate estimated population sizes for EGSS model
#    with Kalman filter, for plotting.
#
#  Choose ML or REML estimates here (by commenting out the unwanted).
#  mu=mu.ml;   ssq=ssq.ml;   tsq=tsq.ml;   x0=x0.ml;
mu=mu.reml;   ssq=ssq.reml;   tsq=tsq.reml;  x0=x0.reml;

m[1]=x0        #  Initial mean of Y(t).
v[1]=tsq       #  Initial variance of Y(t).

for (ti in 1:q) {
                  #  Loop to generate estimated population abundances
                  #    using Kalman filter (see equations 6 & 7,
```

```
                  #    Dennis et al. (2006).
   m[ti+1]=mu+(m[ti]+((v[ti]-tsq)/v[ti])*(Y.t[ti]-m[ti]))
   v[ti+1]=tsq*((v[ti]-tsq)/v[ti])+ssq+tsq
}

#  The following statement calculates exp{E[X(t) | Y(t), Y(t-1),...,Y(0)]};
#    see equation 54 in Dennis et al. (2006).
Predict.t=exp(m+((v-tsq)/v)*(Y.t-m))

#  Plot the data & model-fitted values
plot(Time.t,Observed.t,xlab="time",ylab="population abundance",
   type="o",pch=1,cex=1.5)       #  Population data are circles.
par(lty="dashed")                #  Estimated abundances are dashed line.
points(Time.t,Predict.t, type="l", lwd=1)

#  Print the parameter estimates
parms.egoe=c(mu.egoe,ssq.egoe,tsq.egoe,x0.egoe)  #  Collect for printing
parms.egpn=c(mu.egpn,ssq.egpn,tsq.egpn,x0.egpn)  #          --
parms.reml=c(mu.reml,ssq.reml,tsq.reml,x0.reml)  #          --
parms.ml=c(mu.ml,ssq.ml,tsq.ml,x0.ml)            #          --
names=c("mu","ssq","tsq","x0")                   #          --
types=c("EGOE","EGPN","EGSS-ML","EGSS-REML")     #          --
matrix(cbind(parms.egoe,parms.egpn,parms.ml,parms.reml),
   nrow=4,ncol=4,byrow=TRUE,dimnames=list(types,names))        #  Print stuff

matrix(cbind(mu_lo.reml,mu_hi.reml),nrow=1,ncol=2,byrow=TRUE,
   dimnames=list("95% CI for MU",c("LO","HI")))                #  Print stuff

matrix(cbind(lnlike.ml,AIC.egss),nrow=1,ncol=2,byrow=TRUE,
   dimnames=list("EGSS ML RESULTS",c("LN-LIKELIHOOD","AIC")))  #  Print stuff
```