Appendix - Matlab code

The perturbation analysis results here are obtained using matrix operations, and optimally computed using software designed for matrix calculations. The best such software is MATLAB, which is a nearly universal standard in physics, engineering, and mathematics. The widely used statistical package R can also perform the calculations, although it is not primarily a matrix language. In this section we provide MATLAB code, and in the next we provide the corresponding R code, for the calculations.

Function needed in your working directory, titled `vec.m`

```
function v=vec(x)

    v=x(:);
```

Main code

```
% NEED TO HAVE FILE 'vec.m' in working directory
% INPUT DATA are "q" a vector of death probabilities
% (we used ages 0 to 110+ from HMD),
% and "x", a vector of the average age at death within each age interval,
% (i.e. {a0, 1.5, 2.5, ..., 110.5, 110 + a(omega)} — total length = q+1)


%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% Preliminaries
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

% defining survival probabilities
    p=1—q;

% # of transient states + 1, because in construction of U first row is zero
    s=length(p)+1;

% # transient states only
    s2=length(p);

% other things we will need later
    I = speye(s);   % identity matrix
    e = ones(s,1);  % column vector of ones for summations
    e1 = [1,linspace(0,0,s2)]';

% C matrix is for calculating cumulative sums
    for i=1:s
        C(:,i) = [zeros(i,1);ones(s—i,1)];
    end
```

```
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% Markov chain formulation of longevity
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

% U matrix describes transient states in the Markov chain
    U=sparse(diag(p,-1));
% N matrix, where (i,j) entry is the mean time spent in each age class i,

% conditional upon starting in age class j
    N=inv(I-U);

% M matrix has probability of death at each age on diagonal
    M=sparse(diag(1-p));
    M(s,s)=1;

% The distribution of ages at death
    B=M*N; % the complete distribution of ages at death
    f = B*e1; % the distribution of ages at death from birth (or first age class)

% survivorship (alternatively ell=N*e1)
    ell = e - C*f;

% remaining life expectancy at each age
    mean_eta = sum(N)' - 0.5;

% life expectancy at birth (or first age class)
    eta = e'*N*e1 - 0.5;
% NB: in Markov chain formulations, the life expectancy at birth is always
% 0.5 years higher than that found by conventional life table methods,
% which is why we subtract 0.5 years


%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% Indices of lifespan variation
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

% variance in lifespan
    V=(sum(N)*(2*N-I) - mean_eta'.*mean_eta')';

% standard deviation in lifespan
    S=sqrt(V);

%  e  measure
    dagger_eta=mean_eta'*B;

% Theil's index
    T = f'*((x*eta^(-1)).*(log(x*eta^(-1))));

% Mean Log Deviation
    MLD = f'*(e*log(eta)-log(x));
```

```
% Gini coefficient
    G=1-(1/eta)*e'*(ell.*ell);

% IQR calculations
    F=[cumsum(f)]; % cumulative deaths
    age=[0:s-1]';
    xhat1=interp1(F(1:s-10),age(1:s-10),0.25);

% the 's-10' is in the code to handle zero deaths at oldest ages
    xhat2=interp1(F(1:s-10),age(1:s-10),0.75);
    IQR=xhat2-xhat1;

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% SENSITIVITIES
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

% derivatives of U with respect to mortality change
    for i=1:s-1
        dU_dmu=sparse(zeros(s,s));
        dU_dmu(i+1,i)=-p(i);
        dvecU_dmu(:,i)=vec(dU_dmu);
    end

%derivatives of M with respect to mortality change
    for i=1:s-1
        dM_dmu=sparse(zeros(s,s));
        dM_dmu(i,i)=p(i);
        dvecM_dmu(:,i)=vec(dM_dmu);
    end

%derivative of f with respect to mortality change  %should it be B or N
    df_dmu=kron(e1'*N',I)*dvecM_dmu + kron(e1'*N',B)*dvecU_dmu;

%sensitivity of expected longevity with respect to mortality change
    deta_dmu=kron(e1'*N',e'*N)*dvecU_dmu;

%sensitivity of variance in longevity with respect to mortality change
    dV_dmu  = (2*kron((N')^2,mean_eta')...
            + 2*kron(N',mean_eta'*N)...
            - (I + 2*sparse(diag(mean_eta)))*kron(N',mean_eta'))...
            * dvecU_dmu;

%sensitivity of standard deviation with respect to mortality change
    dS_dmu=0.5*diag(1./S)*dV_dmu;

%sensitivity of eta-dagger with respect to mortality change
    deta_dagger_dmu =(kron(B(:,1)'*N',mean_eta') ...
                    + kron(N(:,1)',mean_eta'*B))*dvecU_dmu ...
                    + kron(N(:,1)',mean_eta')*dvecM_dmu;
```

```
%sensitivity of Theil's with respect to mortality change
    dT_dmu  = (x'*eta^(-1)).*log(x'*eta^(-1))*df_dmu...
            - (T*eta^(-1)+f'*x*eta^(-2))*deta_dmu;

%sensitivity of MLD with respect to mortality change
    dMLD_dmu = (e'*log(eta)-log(x'))*df_dmu+eta^(-1)*deta_dmu;

%sensitivity of Gini with respect to mortality change
    dG_dmu = eta^(-2)*e'*(ell.*ell)*deta_dmu + 2/eta*e'*diag(ell)*C*df_dmu;

% sensitivity of IQR with respect to mortality change
    % sensitivity of cumulative deaths with respect to mortality change
        dF_dmu=[zeros(1,s2);cumsum(df_dmu)];

    % the death density at the quantiles by interpolation
        fxhat1=interp1(age,f,xhat1);
        fxhat2=interp1(age,f,xhat2);
        dFxhat1_dmuT=interp1([0:s]',dF_dmu(:,:),xhat1);
        dFxhat2_dmuT=interp1([0:s]',dF_dmu(:,:),xhat2);

    % finally taking sensitivities
        dxhat_dmu1 = -fxhat1^(-1)*dFxhat1_dmuT;
        dxhat_dmu2 = -fxhat2^(-1)*dFxhat2_dmuT;
        dIQR_dmu=dxhat_dmu2-dxhat_dmu1;


%%%%%%%%%%%%%%%%%%%%%%%%
%ELASTICITIES
%%%%%%%%%%%%%%%%%%%%%%%%

% elasticity of standard deviation
    ES = (1./S(1))*dS_dmu(1,:)*diag(1-p);
% elasticity of variance
    EV = (1./V(1))*dV_dmu(1,:)*diag(1-p);
% elasticity of edagger
    Eedag = (1./dagger_eta(1))'*deta_dagger_dmu*diag(1-p);
% elasticity of Gini
    EG = (1./G(1))*dG_dmu*diag(1-p);
% elasticity of Theil's
    ET = (1./T(1))*dT_dmu*diag(1-p);
% elasticity of MLD
    EMLD = (1./MLD(1))*dMLD_dmu*diag(1-p);
% elasticity of IQR
    EIQR = (1./IQR(1))*dIQR_dmu*diag(1-p);


%%%%%%%%%%%%%%%%%%%%%%%%
%SAVING THE RESULTS
%%%%%%%%%%%%%%%%%%%%%%%%

    res = [stV stedag stS stT stMLD stIQR stG EV Eedag ES ET EMLD EIQR EG];
```

```
res=full(res); % MATLAB can't save sparse arrays
save res.txt res -ascii;
```

Appendix - R code

```
# R code for paper "Perturbation analysis of indices of lifespan variability"
# This code was designed for readability, not efficiency

rm(list = ls(all = TRUE))
options(scipen=6)

#---------------------------------
#------ Input data
#---------------------------------

#-- "q" a vector of death probabilities (we used ages 0 to 110+ from HMD),
#-- and "x", a vector of the average age at death within each age interval
#-- (i.e. a0, 1.5, 2.5 etc., length qi+1)


#---------------------------------
#----- Packages and preliminaries
#---------------------------------


  library(Matrix) # for sparse matrices
  library(signal) # for interp1 function


#---------------------------------
#---- special functions
#---------------------------------

#--- function for making subdiagonals (from Bill Venables)

  subdiag <- function (v, k) {
    n <- length(v) + abs(k)
    x <- matrix(0, n, n)
    if (k == 0)
        diag(x) <- v
    else if (k < 0)
      { ## sub-diagonal
        j <- 1:(n+k)
        i <- (1 - k):n
        x[cbind(i, j)] <- v
      }
```

```
    x
    }


#----------------------------------------
# Preliminaries
#----------------------------------------

# defining survival probabilities
  p <- 1-q

# number of transient states + 1, (in construction of U first row is zero)
  s <- length(p)+1

# number of transient states only
  s2 <- length(p)

# other things we will need later
  I <- diag(rep(1,s))  # identity matrix
  I <- as(I,"sparseMatrix")
  e <- rep(1,s) # vector of ones for summations
  e1 <- c(1,rep(0,s-1))
  age <- 0:s2

# C matrix is for calculating cumulative sums
  C <- Matrix(0,nrow=s,ncol=s)
    for (i in 1:s){
      C[,i] <- c(rep(0,i),rep(1,s-i))
      }
  C <- as(C,"sparseMatrix")

#------------------------------------------------
# Markov chain formulation of longevity
#------------------------------------------------

# U matrix describes transient states in the Markov chain
  U <- subdiag(p,-1)
  U <- as(U,"sparseMatrix")

# N matrix, where (i,j) entry is the mean time spent in each age class i,
# conditional upon starting in age class j
  N <- solve(I-U)
```

```
  N <- as(N,"sparseMatrix")

# M matrix has probability of death at each age on diagonal
  M <- diag(c(q,1))
  M <- as(M,"sparseMatrix")

# The distribution of ages at death
  B <- M %*% N  # the complete distribution of ages at death
  f <- B %*% e1 # the distribution of ages at death from birth (1st age class)
  B <- as(B,"sparseMatrix")

# survivorship (alternatively ell <- e-C%*%f )
  ell <- N %*% e1
  ell <- as(ell,"sparseMatrix")

# remaining life expectancy at each age
  mean_eta <- colSums(N)-0.5

# life expectancy at birth (or first age class)
  eta <- mean_eta[1]

# NB: in Markov chain formulations, the life expectancy at birth is always
# 0.5 years higher than that found by conventional life table methods,
# which is why we subtract 0.5 years



#-----------------------------------
# Indices of lifespan variation
#-----------------------------------

# variance in lifespan
  V <- colSums(N) %*% (2*N-I) - mean_eta*mean_eta

# standard deviation in lifespan
  S <- sqrt(V)

# life disparity (e-dagger)
  eta_dagger <- mean_eta %*% B

# Theil's index
  T <- sum(f*x*eta^(-1)*log(x*eta^(-1)))
```

```
# Mean Log Deviation
  MLD <- sum(f*(e*log(eta)-log(x)))

# Gini coefficient
  G <- 1-(1/eta) * e %*% (ell*ell)

# Interquartile range
  F <- (cumsum(f))
  xhat1 <- interp1(F[1:(s-10)],age[1:(s-10)],0.25)
  xhat2 <- interp1(F[1:(s-10)],age[1:(s-10)],0.75)
  IQR <- xhat2-xhat1
  # the "s-10" is in the code to handle zero deaths at oldest ages

#------------------------------
# SENSITIVITIES
#------------------------------

# derivatives of U with respect to mortality change
   dvecU_dmu <- Matrix(0,nrow=s*s,ncol=s-1)
   r <- seq(2,s*s,s+1) # rows that will contain the elements once stacked
     for (i in 1:(s-1)){
      dvecU_dmu[r[i],i] <- -p[i]
      }
   dvecU_dmu <- as(dvecU_dmu,"sparseMatrix")

# derivatives of M with respect to mortality change
   dvecM_dmu <- Matrix(0,nrow=s*s,ncol=s-1)
   r2 <- seq(1,s*s,s+1) # rows that will contain the elements once stacked
    for (i in 1:(s-1)){
     dvecM_dmu[r2[i],i] <- p[i]
     }
   dvecM_dmu <- as(dvecM_dmu,"sparseMatrix")

# derivative of f with respect to mortality change
   df_dmu <- t(kronecker(N[,1],I)) %*% dvecM_dmu +
     t(kronecker(N[,1],t(B))) %*% dvecU_dmu

# sensitivity of expected longevity with respect to mortality change
   deta_dmu <- t(kronecker(N[,1],colSums(N))) %*% dvecU_dmu

# sensitivity of variance in longevity with respect to mortality change
```

```
   dV_dmu <- (2*kronecker(t(N) %*% t(N),t(mean_eta))  +
     2*kronecker(t(N),mean_eta %*% N) -
     (I + 2*(diag(mean_eta))) %*% kronecker(t(N),t(mean_eta))) %*% dvecU_dmu

# sensitivity of standard deviation with respect to mortality change
   dS_dmu <- 0.5*diag(as.vector(1/S)) %*% dV_dmu

# sensitivity of e-dagger with respect to mortality change
   deta_dagger_dmu <- (kronecker(t(B[,1]) %*% t(N),t(mean_eta)) +
      kronecker(t(N[,1]),(t(mean_eta)%*%B))) %*% dvecU_dmu +
      kronecker(t(N[,1]),t(mean_eta)) %*% dvecM_dmu

# sensitivity of Theil's with respect to mortality change
    dT_dmu <- (t(x) * eta^(-1) * log(t(x) * eta^(-1))) %*% df_dmu -
      (T*eta^(-1) + t(f) %*% x * eta^(-2)) %*% deta_dmu

# sensitivity of MLD with respect to mortality change
    dMLD_dmu <- (t(e) * log(eta) - log(t(x))) %*% df_dmu + eta^(-1) %*% deta_dmu

# sensitivity of Gini with respect to mortality change
    dG_dmu <- eta^(-2) * t(e) %*% (ell*ell) %*% deta_dmu +
      2/eta * t(e) %*% diag(as.vector(ell)) %*% C %*% df_dmu

# sensitivity of IQR with respect to mortality change
    # sensitivity of cumulative deaths with respect to mortality change
        dF_dmu <- matrix(0,nrow=s,ncol=s2)
          for(i in 1:s2){
            dF_dmu[i:s2,i] <- cumsum(df_dmu[i:s2,i])
            }
        dF_dmu <- rbind(rep(0,s2),dF_dmu)

    # the death density at the quantiles by interpolation
        fxhat1 <- interp1(age,f,xhat1)
        fxhat2 <- interp1(age,f,xhat2)

        dFxhat1_dmuT <- rep(0,s2)
        dFxhat2_dmuT <- rep(0,s2)
          for (i in 1:s2){
            dFxhat1_dmuT[i] <- interp1(0:s,dF_dmu[,i],xhat1)
            dFxhat2_dmuT[i] <- interp1(0:s,dF_dmu[,i],xhat2)
            }
```

```
    # finally taking sensitivities
        dxhat_dmu1 <- -fxhat1^(-1)*dFxhat1_dmuT
        dxhat_dmu2 <- -fxhat2^(-1)*dFxhat2_dmuT

    dIQR_dmu <- dxhat_dmu2 - dxhat_dmu1


#------------------------------
# ELASTICITIES
#------------------------------

# elasticity of standard deviation
    ES <- 1/S[1] * dS_dmu[1,] %*% diag(1-p)

# elasticity of variance
    EV <- 1/V[1] * dV_dmu[1,] %*% diag(1-p)

# elasticity of edagger
    EEDAG <- 1/eta_dagger[1] * deta_dagger_dmu %*% diag(1-p)

# elasticity of Gini
    EG <- 1/G[1] * dG_dmu %*% diag(1-p)

# elasticity of Theil's
    ET <- 1/T[1] * dT_dmu %*% diag(1-p)

# elasticity of MLD
    EMLD <- 1/MLD[1] * dMLD_dmu %*% diag(1-p)

# elasticity of IQR
    EIQR <- 1/IQR[1] * dIQR_dmu %*% diag(1-p)


#------------------------------
# PLOTTING THE ELASTICITIES
#------------------------------

  ELAS <- cbind(EEDAG[1,],EG[1,],EIQR[1,],EMLD[1,],ES[1,],ET[1,],EV[1,])
  indices <- c("edag","G","IQR","MLD","S","T","V")
  names(ELAS) <- indices

  matplot(ELAS,xlab="Age",ylab="Elasticity of index",
```

```
#ylim=c(-0.05,0.1),
t="l",lty=1,lwd=2,col=1:7)
legend(legend=indices,"topright",col=1:7,lty=1,lwd=2)
abline(h=0,lty="dotted")
```