# baselineCorr

Documentation of the baselineCorr function.

```
helpFun('baselineCorr')
```

```
Baseline correction of acceleration time history

[COR_XG, COR_XGT, COR_XGTT] = BASELINECORR(T,XGTT)

Description
    Linear baseline correction is performed for an uncorrected
    acceleration time history. Initially, first order fitting (straight
    line) is performed on the acceleration time history and the fitting
    line is subtracted from the acceleration time history, giving thus
    the first correction. Afterwards, the first correction of the
    acceleration is integrated to obtain the velocity, and then first
    order fitting (straight line) is performed on this velocity time
    history. The gradient of the straight fitting line is then subtracted
    from the first correction of the acceleration time history, giving
    thus the second correction of the acceleration time history. The
    second correction of the acceleration time history is then integrated
    to give the corrected velocity and displacement time histories.

Input parameters
    T [double(1:numsteps x 1)] is the time vector of the input
        acceleration time history XGTT. numsteps is the length of the
        input acceleration time history.
    XGTT [double(1:nstep x 1)]: column vector of the acceleration history
        of the excitation imposed at the base. nstep is the number of
        time steps of the dynamic response.

Output parameters
    COR_XG [double(1:nstep x 1)]: time-history of displacement
    COR_XGT [double(1:nstep x 1)]: time-history of velocity
    COR_XGTT [double(1:nstep x 1)]: time-history of acceleration

Example
    %
    fid=fopen('elcentro.dat','r');
    text=textscan(fid,'%f %f');
    fclose(fid);
    time=text{1,1};
    xgtt1=9.81*text{1,2};
    %
    dt=time(2)-time(1);
    %
    xgt1 = cumtrapz(time,xgtt1);
    xg1 = cumtrapz(time,xgt1);
    %
    [xg2, xgt2, xgtt2] = baselineCorr(time,xgtt1)
    %
    figure()
    plot(time,xgtt1)
    hold on
    plot(time,xgtt2)
```

```
%
figure()
plot(time,xgt1)
hold on
plot(time,xgt2)
%
figure()
plot(time,xg1)
hold on
plot(time,xg2)
```


George Papazafeiropoulos
Major, Infrastructure Engineer, Hellenic Air Force
Civil Engineer, M.Sc., Ph.D.
Email: gpapazafeiropoulos@yahoo.gr

*Published with MATLAB® R2017b*

# BLKIN

Documentation of the BLKIN function.

```
helpFun('BLKIN')
```

```
 Bilinear elastoplastic hysteretic model with elastic viscous damping

 [F,K,C,K_STATUS,D] = BLKIN(U,UT,K_HI,K_LO,UY,M,KSI,K_STATUS,D)

 Description
     Define the internal force vector, tangent stiffness matrix and
     tangent damping matrix of a bilinear elastoplastic hysteretic
     structure with elastic damping as a function of displacement and
     velocity.
     The MDOF structure modeled with this function consists of lumped
     masses connected with stiffness and damping elements in series. Each
     lumped mass has one degree of freedom. The first degree of freedom is
     at the top of the structure and the last at its fixed base. However,
     the last degree of freedom is not included in the input arguments of
     the function, i.e. not contained in ndof, as it is always fixed.
     The nonlinear stiffness is virtually of the bilinear type, where an
     initial stiffness and a post-yield stiffness are defined. The
     unloading or reloading curve of this model are parallel to the
     initial loading curve, and a hysteresis loop is created by
     continuously loading and unloading the structure above its yield
     limit. This behavior can be viewed as hardening of the kinematic
     type.
     An appropriate reference for this function definition is Hughes,
     Pister & Taylor (1979): "Implicit-explicit finite elements in
     nonlinear transient analysis". This function should be defined in
     accordance with equations (3.1), (3.2) and (3.3) of this paper. This
     representation has as special cases nonlinear elasticity and a class
     of nonlinear “rate-type” viscoelastic materials. Tangent stiffness
     and tangent damping matrices are the "consistent" linearized
     operators associated to f in the sense of [Hughes & Pister,
     "Consistent linearization in mechanics of solids", Computers and
     Structures, 8 (1978) 391-397].

 Input parameters
     U [double(1 x 1)] is the absolute displacement.
     UT [double(1 x 1)] is the absolute velocity.
     K_HI [double(1 x 1)] is the initial stiffness of the system before
         its first yield, i.e. the high stiffness.
     K_LO [double(1 x 1)] is the post-yield stiffness of the system, i.e.
         the low stiffness.
     UY [double(1 x 1)] is the yield limit of the structure. The structure
         is considered to yield, if the displacement exceeds uy(i).
     M [double(1 x 1)] is the lumped mass.
     KSI [double(1 x 1)] is the ratio of critical viscous damping of the
         system, assumed to be unique for all damping elements of the
         structure.
     K_STATUS [double(1 x 1)] is the is the stiffness vector which takes
         into account any plastic response of the structure. It is used to
         record the status of the structure so that it is known before the
         next application of this function at a next (time) step.
```

```
        Initialize by setting K_STATUS=K_HI.
    D [double(1 x 1)] is the is the equilibrium displacement vector which
        takes into account any plastic response of the structure. It is
        used to record the status of the structure so that it is known
        before the next application of this function at a next (time)
        step. Initialize by setting D=zeros(ndof,1).

Output parameters
    F [double(1 x 1)] is the internal force vector of the structure (sum
        of forces due to stiffness and damping) at displacement u and
        velocity ut
    K [double(1 x 1)] is the tangent stiffness matrix (nonlinear function
        of displacement u and velocity ut). It is equivalent to the
        derivative d(f)/d(u)
    C [double(1 x 1)] is the tangent damping matrix (nonlinear function
        of displacement u and velocity ut). It is equivalent to the
        derivative d(f)/d(u)
    K_STATUS [double(1 x 1)] is the is the stiffness vector which takes
        into account any plastic response of the structure. It is used to
        record the status of the structure so that it is known before the
        next application of this function at a next (time) step.
    D [double(1 x 1)] is the is the equilibrium displacement vector which
        takes into account any plastic response of the structure. It is
        used to record the status of the structure so that it is known
        before the next application of this function at a next (time)
        step.

Example
    %
    u=0:0.2:4;
    u=[u,u(end:-1:1)];
    u=[u,-u];
    u=[u u];
    %
    ut=0.001*ones(1,numel(u));
    ut=[ut,-ut];
    ut=[ut,ut(end:-1:1)];
    ut=[ut ut];
    %
    k_hi=1000;
    %
    k_lo=1;
    %
    uy=2;
    %
    M=1;
    %
    ksi=0.05;
    %
    k=k_hi;
    %
    d=0;
    %
    f=zeros(1,numel(u));
    for i=1:numel(u)
        [f(i),K,C,k,d] = BLKIN(u(i),ut(i),k_hi,k_lo,uy,M,ksi,k,d);
    end
    %
    figure()
    plot(u,f)
```

George Papazafeiropoulos
Major, Infrastructure Engineer, Hellenic Air Force
Civil Engineer, M.Sc., Ph.D.
Email: gpapazafeiropoulos@yahoo.gr

Published with MATLAB® R2017b

# CDReSp

Documentation of the CDReSp function.

```
helpFun('CDReSp')
```

```
  Constant Ductility Response Spectra (CDReSp)

  [PSA,PSV,SD,SV,SA,FYK,MUK,ITERK]=CDRESP(DT,XGTT,T,KSI,MU,N,TOL,REDF,DTTOL,ALGID,RINF)

  Description
      The constant ductility response spectra for a given time-history of
      constant time step, a given eigenperiod range, a given viscous
      damping ratio and a given ductility are computed. See section 7.5 in
      Chopra (2012).

  Input parameters
      DT [double(1 x 1)] is the time step of the input acceleration time
          history XGTT.
      XGTT [double(1:numsteps x 1)] is the input acceleration time history.
          numsteps is the length of the input acceleration time history.
      T [double(1:numSDOFs x 1)] contains the values of eigenperiods for
          which the response spectra are requested. numSDOFs is the number
          of SDOF oscillators being analysed to produce the spectra.
      KSI [double(1 x 1)] is the fraction of critical viscous damping.
      MU [double(1 x 1)] is the target ductility for which the response
          spectra are calculated.
      N [double(1 x 1)] is the maximum number of iterations that can be
          performed until convergence of the calculated ductility to the
          target ductility is achieved. Default value 50.
      TOL [double(1 x 1)] is the tolerance for convergence for the target
          ductility. Default value 0.01.
      REDF [double(1 x 1)] is the reduction factor of the lower bound of
          the range of the yield limit. The yield limit that corresponds to
          to ductility of the SDOF system equal to the target ductility is
          assumed to be in the range [maxU/REDF,maxU], where maxU is the
          maximum (absolute) displacement of the linear equivalent SDOF
          system. Increasing REDF reduces the lower limit increasing thus
          the available range. Default value 4.
      DTTOL [double(1 x 1)] is the tolerance for resampling of the input
          acceleration time history. For a given eigenperiod T, resampling
          takes place if DT/T>dtTol. Default value 0.01.
      ALGID [char(1 x :inf)] is the algorithm to be used for the time
          integration. It can be one of the following strings for superior
          optimally designed algorithms:
              'generalized a-method': The generalized a-method (Chung &
              Hulbert, 1993)
              'HHT a-method': The Hilber-Hughes-Taylor method (Hilber,
              Hughes & Taylor, 1977)
              'WBZ': The Wood–Bossak–Zienkiewicz method (Wood, Bossak &
              Zienkiewicz, 1980)
              'U0-V0-Opt': Optimal numerical dissipation and dispersion
              zero order displacement zero order velocity algorithm
              'U0-V0-CA': Continuous acceleration (zero spurious root at
              the low frequency limit) zero order displacement zero order
              velocity algorithm
```

```
            'U0-V0-DA': Discontinuous acceleration (zero spurious root at
            the high frequency limit) zero order displacement zero order
            velocity algorithm
            'U0-V1-Opt': Optimal numerical dissipation and dispersion
            zero order displacement first order velocity algorithm
            'U0-V1-CA': Continuous acceleration (zero spurious root at
            the low frequency limit) zero order displacement first order
            velocity algorithm
            'U0-V1-DA': Discontinuous acceleration (zero spurious root at
            the high frequency limit) zero order displacement first order
            velocity algorithm
            'U1-V0-Opt': Optimal numerical dissipation and dispersion
            first order displacement zero order velocity algorithm
            'U1-V0-CA': Continuous acceleration (zero spurious root at
            the low frequency limit) first order displacement zero order
            velocity algorithm
            'U1-V0-DA': Discontinuous acceleration (zero spurious root at
            the high frequency limit) first order displacement zero order
            velocity algorithm
            'Newmark ACA': Newmark Average Constant Acceleration method
            'Newmark LA': Newmark Linear Acceleration method
            'Newmark BA': Newmark Backward Acceleration method
            'Fox-Goodwin': Fox-Goodwin formula
        Default value 'U0-V0-CA'.
    RINF [double(1 x 1)] is the minimum absolute value of the eigenvalues
        of the amplification matrix. For the amplification matrix see
        eq.(61) in Zhou & Tamma (2004). Default value 0.

Output parameters
    PSA [double(1:numSDOFs x 1)] is the Pseudo-Spectral Acceleration.
    PSV [double(1:numSDOFs x 1)] is the Pseudo-Spectral Velocity.
    SD [double(1:numSDOFs x 1)] is the Spectral Displacement.
    SV [double(1:numSDOFs x 1)] is the Spectral Velocity.
    SA [double(1:numSDOFs x 1)] is the Spectral Acceleration.
    FYK [double(1:numSDOFs x 1)] is the yield limit that each SDOF must
        have in order to attain ductility equal to muK.
    MUK [double(1:numSDOFs x 1)] is the achieved ductility for each
        period (each SDOF).
    ITERK [double(1:numSDOFs x 1)] is the number of iterations needed for
        convergence for each period (each SDOF).

Example
    %
    dt=0.02;
    %
    N=10;
    a=rand(N,1)-0.5;
    b=100*pi*rand(N,1);
    c=pi*(rand(N,1)-0.5);
    t=(0:dt:(100*dt))';
    xgtt=zeros(size(t));
    for i=1:N
        xgtt=xgtt+a(i)*sin(b(i)*t+c(i));
    end
    %
    T=(0.04:0.04:4)';
    %
    ksi=0.05;
    %
    mu=2;
    %
```

```
n=50;
%
tol=0.01;
%
redf=4;
%
dtTol=0.02;
%
AlgID='U0-V0-Opt';
%
rinf=1;
%
[CDPSa,CDPSv,CDSd,CDSv,CDSa,fyK,muK,iterK]=CDReSp(dt,xgtt,T,ksi,...
    mu,n,tol,redf,dtTol,AlgID,rinf);
%
plot(T,CDSd)
```

---


George Papazafeiropoulos
Major, Infrastructure Engineer, Hellenic Air Force
Civil Engineer, M.Sc., Ph.D.
Email: gpapazafeiropoulos@yahoo.gr

---

*Published with MATLAB® R2017b*

# DRHA

Documentation of the DRHA function.

```
helpFun('DRHA')
```

```
  Dynamic Response History Analysis (DRHA) of a SDOF system

  [U,V,A,F,ES,ED] = DRHA(K,M,DT,XGTT,KSI,U0,UT0,RINF)

  Description
      Determine the time history of structural response of a SDOF system

  Input parameters
      K [double(1 x 1)] is the stiffness of the system.
      M [double(1 x 1)] is the lumped masses of the structure.
      DT [double(1 x 1)] is the time step of the response history analysis
          from which the response spectrum is calculated
      XGTT [double(1:nstep x 1)]: column vector of the acceleration history
          of the excitation imposed at the base. nstep is the number of
          time steps of the dynamic response.
      KSI [double(1 x 1)] is the ratio of critical damping of the SDOF
          system. Default value 0.05.
      U0 [double(1 x 1)] is the initial displacement of the SDOF system.
          Default value 0.
      UT0 [double(1 x 1)] is the initial velocity of the SDOF system.
          Default value 0.
      RINF [double(1 x 1)] is the minimum absolute value of the eigenvalues
          of the amplification matrix. For the amplification matrix see
          eq.(61) in Zhou & Tamma (2004). Default value 1.

  Output parameters
      U [double(1 x 1:nstep)]: displacement time history.
      V [double(1 x 1:nstep)]: velocity time history.
      A [double(1 x 1:nstep)]: acceleration time history.
      F [double(1 x 1:nstep)]: equivalent static force time history.
      ES [double(1 x 1:nstep)]: time-history of the recoverable
          strain energy of the system (total and not incremental).
      ED [double(1 x 1:nstep)]: time-history of the energy
          dissipated by viscoelastic damping during each time step
          (incremental). cumsum(Ed) gives the time history of the total
          energy dissipated at dof i from the start of the dynamic
          analysis.

 __________________________________________________________________________
  Copyright (c) 2018-2022
      George Papazafeiropoulos
      Major, Infrastructure Engineer, Hellenic Air Force
      Civil Engineer, M.Sc., Ph.D.
      Email: gpapazafeiropoulos@yahoo.gr
  _________________________________________________________________________
```

# FASp

Documentation of the FASp function.

```
helpFun('FASp')
```

```
Single sided Fourier amplitude spectrum

[F,U] = FASP(DT,XGTT)

Description
    Fourier amplitude spectrum of an earthquake.

Input parameters
    DT [double(1 x 1)] is the time step of the input acceleration time
        history XGTT.
    XGTT [double(1:numsteps x 1)] is the input acceleration time history.
        numsteps is the length of the input acceleration time history.

Output parameters
    F [double(1:2^(nextpow2(length(XGTT))-1) x 1)] is the frequency range in
        which the Fourier amplitudes are calculated.
    U [double(1:2^(nextpow2(length(XGTT))-1) x 1)] contains the Fourier amplitudes

__________________________________________________________________________
Copyright (c) 2018-2022
    George Papazafeiropoulos
    Major, Infrastructure Engineer, Hellenic Air Force
    Civil Engineer, M.Sc., Ph.D.
    Email: gpapazafeiropoulos@yahoo.gr
 _________________________________________________________________________
```

# HalfStep

Documentation of the HalfStep function.

```
helpFun('HalfStep')
```

```
 Reproduce signal with half time step
 
 UNEW = HALFSTEP(U)
 
 Input parameters
     U [double(1:n x 1)] is the input signal with time step dt.
 
 Output parameters
     UNEW [double(1:n x 1)] is the output signal with time step dt/2.
 
 Verification:
     %
     u=0.2:0.2:4;
     %
     uNew=HalfStep(u);
     %
     figure()
     plot((1:numel(u)),u)
     hold on
     plot((1:0.5:numel(u)),uNew)
```



George Papazafeiropoulos
Major, Infrastructure Engineer, Hellenic Air Force
Civil Engineer, M.Sc., Ph.D.
Email: gpapazafeiropoulos@yahoo.gr


__________________________________________________________________________

*Published with MATLAB® R2017b*

## LEReSp

Documentation of the LEReSp function.

```
helpFun('LEReSp')
```

```
 Fast calculation of Linear Elastic Response Spectra (LEReSp) and
 pseudospectra.

 [PSA,PSV,SD,SV,SA,SIEVABS,SIEVREL]=LERESP(DT,XGTT,T,KSI)

 Description
     The linear elastic response spectra for a given time-history of
     constant time step, a given eigenperiod range and a given viscous
     damping ratio are computed.

 Input parameters
     DT [double(1 x 1)] is the time step of the input acceleration time
         history XGTT.
     XGTT [double(1:numsteps x 1)] is the input acceleration time history.
         numsteps is the length of the input acceleration time history.
     T [double(1:numSDOFs x 1)] contains the values of eigenperiods for
         which the response spectra are requested. numSDOFs is the number
         of SDOF oscillators being analysed to produce the spectra.
     KSI [double(1 x 1)] is the fraction of critical viscous damping.
     DTTOL [double(1 x 1)] is the maximum ratio of the integration time
         step to the eigenperiod. Default value 0.01.
     ALGID [char(1 x :inf)] is the algorithm to be used for the time
         integration. It can be one of the following strings for superior
         optimally designed algorithms:
             'generalized a-method': The generalized a-method (Chung &
             Hulbert, 1993)
             'HHT a-method': The Hilber-Hughes-Taylor method (Hilber,
             Hughes & Taylor, 1977)
             'WBZ': The Wood–Bossak–Zienkiewicz method (Wood, Bossak &
             Zienkiewicz, 1980)
             'U0-V0-Opt': Optimal numerical dissipation and dispersion
             zero order displacement zero order velocity algorithm
             'U0-V0-CA': Continuous acceleration (zero spurious root at
             the low frequency limit) zero order displacement zero order
             velocity algorithm
             'U0-V0-DA': Discontinuous acceleration (zero spurious root at
             the high frequency limit) zero order displacement zero order
             velocity algorithm
             'U0-V1-Opt': Optimal numerical dissipation and dispersion
             zero order displacement first order velocity algorithm
             'U0-V1-CA': Continuous acceleration (zero spurious root at
             the low frequency limit) zero order displacement first order
             velocity algorithm
             'U0-V1-DA': Discontinuous acceleration (zero spurious root at
             the high frequency limit) zero order displacement first order
             velocity algorithm
             'U1-V0-Opt': Optimal numerical dissipation and dispersion
             first order displacement zero order velocity algorithm
             'U1-V0-CA': Continuous acceleration (zero spurious root at
             the low frequency limit) first order displacement zero order
```

```
            velocity algorithm
            'U1-V0-DA': Discontinuous acceleration (zero spurious root at
            the high frequency limit) first order displacement zero order
            velocity algorithm
            'Newmark ACA': Newmark Average Constant Acceleration method
            'Newmark LA': Newmark Linear Acceleration method
            'Newmark BA': Newmark Backward Acceleration method
            'Fox-Goodwin': Fox-Goodwin formula
    RINF [double(1 x 1)] is the minimum absolute value of the eigenvalues
        of the amplification matrix. For the amplification matrix see
        eq.(61) in Zhou & Tamma (2004).

Output parameters
    PSA [double(1:numSDOFs x 1)] is the Pseudo Acceleration Spectrum.
    PSV [double(1:numSDOFs x 1)] is the Pseudo Velocity Spectrum.
    SD [double(1:numSDOFs x 1)] is the Spectral Displacement.
    SV [double(1:numSDOFs x 1)] is the Spectral Velocity.
    SA [double(1:numSDOFs x 1)] is the Spectral Acceleration.
    SIEVABS [double(1:numSDOFs x 1)] is the equivalent absolute input
        energy velocity.
    SIEVREL [double(1:numSDOFs x 1)] is the equivalent relative input
        energy velocity.

Example
    %
    dt=0.02;
    %
    N=10;
    a=rand(N,1)-0.5;
    b=100*pi*rand(N,1);
    c=pi*(rand(N,1)-0.5);
    t=(0:dt:(100*dt))';
    xgtt=zeros(size(t));
    for i=1:N
        xgtt=xgtt+a(i)*sin(b(i)*t+c(i));
    end
    %
    T=logspace(log10(0.02),log10(50),1000)';
    %
    ksi=0.05;
    %
    [PSa,PSv,Sd,Sv,Sa,SievABS,SievREL]=LEReSp(dt,xgtt,T,ksi);
```

---



```
    George Papazafeiropoulos
    Major, Infrastructure Engineer, Hellenic Air Force
    Civil Engineer, M.Sc., Ph.D.
    Email: gpapazafeiropoulos@yahoo.gr
```


---

# LIDA

Documentation of the LIDA function.

```
helpFun('LIDA')
```

```
Linear Implicit Dynamic Analysis (LIDA)

[U,UT,UTT] = LIDA(DT,XGTT,OMEGA,KSI,U0,UT0,RINF)

Description
    Linear implicit direct time integration of second order differential
    equation of motion of dynamic response of linear elastic SDOF systems
    The General Single Step Single Solve (GSSSS) family of algorithms
    published by X.Zhou & K.K.Tamma (2004) is employed for direct time
    integration of the general linear or nonlinear structural Single
    Degree of Freedom (SDOF) dynamic problem. The optimal numerical
    dissipation and dispersion zero order displacement zero order
    velocity algorithm designed according to the above journal article,
    is used in this routine. This algorithm encompasses the scope of
    Linear Multi-Step (LMS) methods and is limited by the Dahlquist
    barrier theorem (Dahlquist,1963). The force - displacement - velocity
    relation of the SDOF structure is linear.

Input parameters
    DT [double(1 x 1)] is the time step
    XGTT [double(1:nstep x 1)] is the column vector of the acceleration
        history of the excitation imposed at the base. nstep is the
        number of time steps of the dynamic response.
    OMEGA [double(1 x 1)] is the eigenfrequency of the structure in
        rad/sec.
    KSI [double(1 x 1)] is the ratio of critical damping of the SDOF
        system. Default value 0.05.
    U0 [double(1 x 1)] is the initial displacement of the SDOF system.
        Default value 0.
    UT0 [double(1 x 1)] is the initial velocity of the SDOF system.
        Default value 0.
    ALGID [char(1 x :inf)] is the algorithm to be used for the time
        integration. It can be one of the following strings for superior
        optimally designed algorithms:
            'generalized a-method': The generalized a-method (Chung &
            Hulbert, 1993)
            'HHT a-method': The Hilber-Hughes-Taylor method (Hilber,
            Hughes & Taylor, 1977)
            'WBZ': The Wood–Bossak–Zienkiewicz method (Wood, Bossak &
            Zienkiewicz, 1980)
            'U0-V0-Opt': Optimal numerical dissipation and dispersion
            zero order displacement zero order velocity algorithm
            'U0-V0-CA': Continuous acceleration (zero spurious root at
            the low frequency limit) zero order displacement zero order
            velocity algorithm
            'U0-V0-DA': Discontinuous acceleration (zero spurious root at
            the high frequency limit) zero order displacement zero order
            velocity algorithm
            'U0-V1-Opt': Optimal numerical dissipation and dispersion
            zero order displacement first order velocity algorithm
```

```
            'U0-V1-CA': Continuous acceleration (zero spurious root at
            the low frequency limit) zero order displacement first order
            velocity algorithm
            'U0-V1-DA': Discontinuous acceleration (zero spurious root at
            the high frequency limit) zero order displacement first order
            velocity algorithm
            'U1-V0-Opt': Optimal numerical dissipation and dispersion
            first order displacement zero order velocity algorithm
            'U1-V0-CA': Continuous acceleration (zero spurious root at
            the low frequency limit) first order displacement zero order
            velocity algorithm
            'U1-V0-DA': Discontinuous acceleration (zero spurious root at
            the high frequency limit) first order displacement zero order
            velocity algorithm
            'Newmark ACA': Newmark Average Constant Acceleration method
            'Newmark LA': Newmark Linear Acceleration method
            'Newmark BA': Newmark Backward Acceleration method
            'Fox-Goodwin': Fox-Goodwin formula
    RINF [double(1 x 1)] is the minimum absolute value of the eigenvalues
        of the amplification matrix. For the amplification matrix see
        eq.(61) in Zhou & Tamma (2004). Default value 1.

Output parameters
    U [double(1:nstep x 1)] is the time-history of displacement
    UT [double(1:nstep x 1)] is the time-history of velocity
    UTT [double(1:nstep x 1)] is the time-history of acceleration

Example (Figure 6.6.1 in Chopra, Tn=1sec)
    dt=0.02;
    fid=fopen('elcentro.dat','r');
    text=textscan(fid,'%f %f');
    fclose(fid);
    xgtt=9.81*text{1,2};
    Tn=1;
    omega=2*pi/Tn;
    ksi=0.02;
    u0=0;
    ut0=0;
    rinf=1;
    [u,ut,utt] = LIDA(dt,xgtt,omega,ksi,u0,ut0,rinf);
    D=max(abs(u))/0.0254
```

---


George Papazafeiropoulos
Major, Infrastructure Engineer, Hellenic Air Force
Civil Engineer, M.Sc., Ph.D.
Email: gpapazafeiropoulos@yahoo.gr

---

# NLIDABLKIN

Documentation of the NLIDABLKIN function.

```
helpFun('NLIDABLKIN')
```

```
Non Linear Implicit Dynamic Analysis of a bilinear kinematic hardening
hysteretic structure with elastic damping

[U,UT,UTT,FS,EY,ES,ED,JITER] = NLIDABLKIN(DT,XGTT,M,K_HI,K_LO,UY,KSI,...
    ALGID,U0,UT0,RINF,MAXTOL,JMAX,DAK)

Description
    General linear implicit direct time integration of second order
    differential equations of a bilinear elastoplastic hysteretic SDOF
    dynamic system with elastic damping, with lumped mass.
    The General Single Step Single Solve (GSSSS) family of algorithms
    published by X.Zhou & K.K.Tamma (2004) is employed for direct time
    integration of the general linear or nonlinear structural Single
    Degree of Freedom (SDOF) dynamic problem. Selection among 9
    algorithms, all designed according to the above journal article, can
    be made in this routine. These algorithms encompass the scope of
    Linear Multi-Step (LMS) methods and are limited by the Dahlquist
    barrier theorem (Dahlquist,1963).

Input parameters
    DT [double(1 x 1)] is the time step of the integration
    XGTT [double(1:NumSteps x 1)] is the acceleration time history which
        is imposed at the lumped mass of the SDOF structure.
    M [double(1 x 1)] is the lumped masses of the structure. Define the
        lumped masses from the top to the bottom, excluding the fixed dof
        at the base
    K_HI [double(1 x 1)] is the initial stiffness of the system before
        its first yield, i.e. the high stiffness. Give the stiffness of
        each storey from top to bottom.
    K_LO [double(1 x 1)] is the post-yield stiffness of the system,
        i.e. the low stiffness. Give the stiffness of each storey from
        top to bottom.
    UY [double(1 x 1)] is the yield limit of the stiffness elements of
        the structure. The element is considered to yield, if the
        interstorey drift between degrees of freedom i and i+1 exceeds
        UY(i). Give the yield limit of each storey from top to bottom.
    KSI [double(1 x 1)] is the ratio of critical viscous damping of the
        system, assumed to be unique for all damping elements of the
        structure.
    ALGID [char(1 x :inf)] is the algorithm to be used for the time
        integration. It can be one of the following strings for superior
        optimally designed algorithms:
            'generalized a-method': The generalized a-method (Chung &
            Hulbert, 1993)
            'HHT a-method': The Hilber-Hughes-Taylor method (Hilber,
            Hughes & Taylor, 1977)
            'WBZ': The Wood–Bossak–Zienkiewicz method (Wood, Bossak &
            Zienkiewicz, 1980)
            'U0-V0-Opt': Optimal numerical dissipation and dispersion
            zero order displacement zero order velocity algorithm
```

```
        'U0-V0-CA': Continuous acceleration (zero spurious root at
        the low frequency limit) zero order displacement zero order
        velocity algorithm
        'U0-V0-DA': Discontinuous acceleration (zero spurious root at
        the high frequency limit) zero order displacement zero order
        velocity algorithm
        'U0-V1-Opt': Optimal numerical dissipation and dispersion
        zero order displacement first order velocity algorithm
        'U0-V1-CA': Continuous acceleration (zero spurious root at
        the low frequency limit) zero order displacement first order
        velocity algorithm
        'U0-V1-DA': Discontinuous acceleration (zero spurious root at
        the high frequency limit) zero order displacement first order
        velocity algorithm
        'U1-V0-Opt': Optimal numerical dissipation and dispersion
        first order displacement zero order velocity algorithm
        'U1-V0-CA': Continuous acceleration (zero spurious root at
        the low frequency limit) first order displacement zero order
        velocity algorithm
        'U1-V0-DA': Discontinuous acceleration (zero spurious root at
        the high frequency limit) first order displacement zero order
        velocity algorithm
        'Newmark ACA': Newmark Average Constant Acceleration method
        'Newmark LA': Newmark Linear Acceleration method
        'Newmark BA': Newmark Backward Acceleration method
        'Fox-Goodwin': Fox-Goodwin formula
    U0 [double(1 x 1)] is the initial displacement.
    UT0 [double(1 x 1)] is the initial velocity.
    RINF [double(1 x 1)] is the minimum absolute value of the eigenvalues
        of the amplification matrix. For the amplification matrix see
        eq.(61) in Zhou & Tamma (2004).
    MAXTOL [double(1 x 1)] is the maximum tolerance of convergence of the
        Full Newton Raphson method for numerical computation of
        acceleration.
    JMAX [double(1 x 1)] is the maximum number of iterations per
        increment. If JMAX=0 then iterations are not performed and the
        MAXTOL parameter is not taken into account.
    DAK [double(1 x 1)] is the infinitesimal acceleration for the
        calculation of the derivetive required for the convergence of the
        Newton-Raphson iteration.

Output parameters
    U [double(1 x 1:NumSteps)] is the time-history of displacement
    UT [double(1 x 1:NumSteps)] is the time-history of velocity
    UTT [double(1 x 1:NumSteps)] is the time-history of acceleration
    FS [double(1 x 1:NumSteps)] is the time-history of the internal
        force of the structure analysed.
    EY [double(1 x 1:NumSteps)] is the time history of the sum of the
        energy dissipated by yielding during each time step and the
        recoverable strain energy of the system (incremental).
        cumsum(EY)-Es gives the time history of the total energy
        dissipated by yielding from the start of the dynamic analysis.
    ES [double(1 x 1:NumSteps)] is the time-history of the recoverable
        strain energy of the system (total and not incremental).
    ED [double(1 x 1:NumSteps)] is the time-history of the energy
        dissipated by viscoelastic damping during each time step
        (incremental). cumsum(ED) gives the time history of the total
        energy dissipated from the start of the dynamic analysis.
    JITER [double(1 x 1:NumSteps)] is the iterations per increment

Notation in the code
```

```
    u=displacement
    un=displacement after increment n
    ut=velocity
    utn=velocity after increment n
    utt=acceleration
    uttn=acceleration after increment n
```


George Papazafeiropoulos
Major, Infrastructure Engineer, Hellenic Air Force
Civil Engineer, M.Sc., Ph.D.
Email: gpapazafeiropoulos@yahoo.gr

*Published with MATLAB® R2017b*

# OpenSeismoMatlab

Documentation of the OpenSeismoMatlab function.

```
helpFun('OpenSeismoMatlab')
```

```
 Seismic parameters of an acceleration time history

 PARAM=OPENSEISMOMATLAB(DT,XGTT,SW,BASELINESW,DTI,KSI,T,MU,ALGID)

 Description
     This function calculates the seismic parameters from an acceleration
     time history. More specifically, it calculates the following:
     1) Velocity vs time
     2) Displacement vs time
     3) Resampled acceleration time history (i.e. the input acceleration
     time history with modified time step size)
     4) Peak ground acceleration
     5) Peak ground velocity
     6) Peak ground displacement
     7) Total cumulative energy and normalized cumulative energy vs time
     8) Significant duration according to Trifunac & Brady (1975)
     9) Total Arias intensity (Ia)
     10) Linear elastic pseudo-acceleration response spectrum
     11) Linear elastic pseudo-velocity response spectrum
     12) Linear elastic displacement response spectrum
     13) Linear elastic velocity response spectrum
     14) Linear elastic acceleration response spectrum
     15) Constant ductility displacement response spectrum
     16) Constant ductility velocity response spectrum
     17) Constant ductility acceleration response spectrum
     18) Fourier amplitude spectrum
     19) Mean period (Tm)

 Input parameters
     DT [double(1 x 1)] is the size of the time step of the input
         acceleration time history xgtt.
     XGTT [double(1:numsteps x 1)] is the input acceleration time history.
     SW [char(1 x :inf)] is a string which determines which parameters of
         the input acceleration time history will be calculated. sw can
         take one of the following values (strings are case insensitive):
         'TIMEHIST': the displacement, velocity and acceleration time
             histories are calculated.
         'RESAMPLE': the acceleration time history with modified time step
             size is calculated.
         'PGA': The peak ground acceleration is calculated.
         'PGV': The peak ground velocity is calculated.
         'PGD': The peak ground displacement is calculated.
         'ARIAS': The total cumulative energy, significant duration
             according to Trifunac & Brady (1975) and Arias intensity are
             calculated.
         'ES': The linear elastic response spectra and pseudospectra are
             calculated.
         'CDS': The constant ductility response spectra are calculated.
         'FAS': The Fourier amplitude spectrum and the mean period are
             calculated.
```

```
    BASELINESW [logical(1 x 1)] determines if baseline correction will be
        applied for the calculation of the various output quantities.
    DTI [double(1 x 1)] is the new time step size for resampling of the
        input acceleration time history.
    KSI [double(1 x 1)] is the fraction of critical viscous damping.
    T [double(1:numSDOFs x 1)] contains the values of eigenperiods for
        which the response spectra are requested. numSDOFs is the number
        of SDOF oscillators being analysed to produce the spectra.
    MU [double(1 x 1)] is the specified ductility for which the response
        spectra are calculated.
    ALGID [char(1 x :inf)] is the algorithm to be used for the time
        integration, if applicable. It can be one of the following
        strings for superior optimally designed algorithms (strings are
        case sensitive):
            'generalized a-method': The generalized a-method (Chung &
            Hulbert, 1993)
            'HHT a-method': The Hilber-Hughes-Taylor method (Hilber,
            Hughes & Taylor, 1977)
            'WBZ': The Wood–Bossak–Zienkiewicz method (Wood, Bossak &
            Zienkiewicz, 1980)
            'U0-V0-Opt': Optimal numerical dissipation and dispersion
            zero order displacement zero order velocity algorithm
            'U0-V0-CA': Continuous acceleration (zero spurious root at
            the low frequency limit) zero order displacement zero order
            velocity algorithm
            'U0-V0-DA': Discontinuous acceleration (zero spurious root at
            the high frequency limit) zero order displacement zero order
            velocity algorithm
            'U0-V1-Opt': Optimal numerical dissipation and dispersion
            zero order displacement first order velocity algorithm
            'U0-V1-CA': Continuous acceleration (zero spurious root at
            the low frequency limit) zero order displacement first order
            velocity algorithm
            'U0-V1-DA': Discontinuous acceleration (zero spurious root at
            the high frequency limit) zero order displacement first order
            velocity algorithm
            'U1-V0-Opt': Optimal numerical dissipation and dispersion
            first order displacement zero order velocity algorithm
            'U1-V0-CA': Continuous acceleration (zero spurious root at
            the low frequency limit) first order displacement zero order
            velocity algorithm
            'U1-V0-DA': Discontinuous acceleration (zero spurious root at
            the high frequency limit) first order displacement zero order
            velocity algorithm
            'Newmark ACA': Newmark Average Constant Acceleration method
            'Newmark LA': Newmark Linear Acceleration method
            'Newmark BA': Newmark Backward Acceleration method
            'Fox-Goodwin': Fox-Goodwin formula

Output parameters
    PARAM (structure) has the following fields:
        PARAM.vel [double(1:numsteps x 1)] Velocity vs time
        PARAM.disp [double(1:numsteps x 1)] Displacement vs time
        PARAM.PGA [double(1 x 1)] Peak ground acceleration
        PARAM.PGV [double(1 x 1)] Peak ground velocity
        PARAM.PGD [double(1 x 1)] Peak ground displacement
        PARAM.Ecum [double(1 x 1)] Total cumulative energy
        PARAM.EcumTH [double(1:numsteps x 1)] normalized cumulative
        energy vs time
        PARAM.t_5_95 [double(1 x 2)] Time instants at which 5% and 95% of
        cumulative energy have occurred
```

```
PARAM.Td [double(1 x 1)] Time between when 5% and 95% of
cumulative energy has occurred (significant duration according to
Trifunac-Brady (1975))
PARAM.arias [double(1 x 1)] Total Arias intensity (Ia)
PARAM.PSa [double(:inf x 1)] Linear elastic pseudo-acceleration
response spectrum
PARAM.PSv [double(:inf x 1)] Linear elastic pseudo-velocity
response spectrum
PARAM.Sd [double(:inf x 1)] Linear elastic displacement response
spectrum
PARAM.Sv [double(:inf x 1)] Linear elastic velocity response
spectrum
PARAM.Sa [double(:inf x 1)] Linear elastic acceleration response
spectrum
PARAM.SievABS [double(:inf x 1)] Linear elastic absolute input
energy equivalent velocity spectrum
PARAM.SievREL [double(:inf x 1)] Linear elastic relative input
energy equivalent velocity spectrum
PARAM.PredPSa [double(1 x 1)] Predominant acceleration of the PSa
spectrum
PARAM.PredPeriod [double(1 x 1)] Predominant period of the PSa
spectrum
PARAM.CDPSa [double(:inf x 1)] Constant ductility
pseudo-acceleration response spectrum
PARAM.CDPSv [double(:inf x 1)] Constant ductility pseudo-velocity
response spectrum
PARAM.CDSd [double(:inf x 1)] Constant ductility displacement
response spectrum
PARAM.CDSv [double(:inf x 1)] Constant ductility velocity
response spectrum
PARAM.CDSa [double(:inf x 1)] Constant ductility acceleration
response spectrum
PARAM.fyK [double(:inf x 1)] yield limit that each SDOF must have
in order to attain ductility equal to PARAM.muK.
PARAM.muK [double(:inf x 1)] achieved ductility for each period
(each SDOF).
PARAM.iterK [double(:inf x 1)] number of iterations needed for
convergence for each period (each SDOF).
PARAM.FAS [double(1:2^(nextpow2(length(xgtt))-1) x 1)] Fourier
amplitude spectrum
PARAM.Tm [double(1 x 1)] Mean period (Tm)
PARAM.Fm [double(1 x 1)] Mean frequency (Fm)
```


George Papazafeiropoulos
Major, Infrastructure Engineer, Hellenic Air Force
Civil Engineer, M.Sc., Ph.D.
Email: gpapazafeiropoulos@yahoo.gr

# Constant ductility response spectra

Generate the constant ductility response spectra and associated results of an earthquake suite using OpenSeismoMatlab.

## Contents



## Input

earthquake motions

```
eqmotions={'Imperial Valley'; % Imperial valley 1979
    'Kocaeli';
    'Loma Prieta';
    'Northridge';
    'San Fernando';
    'Spitak';
    'Cape Mendocino';
    'ChiChi';
    'El Centro'; % Imperial valley 1940
    'Hollister';
    'Kobe'};
```

Set the eigenperiod range for which the response spectra will be calculated.

```
Tspectra=(0.08:0.08:4)';
```

Set critical damping ratio of the response spectra to be calculated.

```
ksi=0.05;
```

Set the target ductility (not used here)

```
mu=2;
```

Extract nonlinear response spectra

```
sw='cds';
```

## Calculation

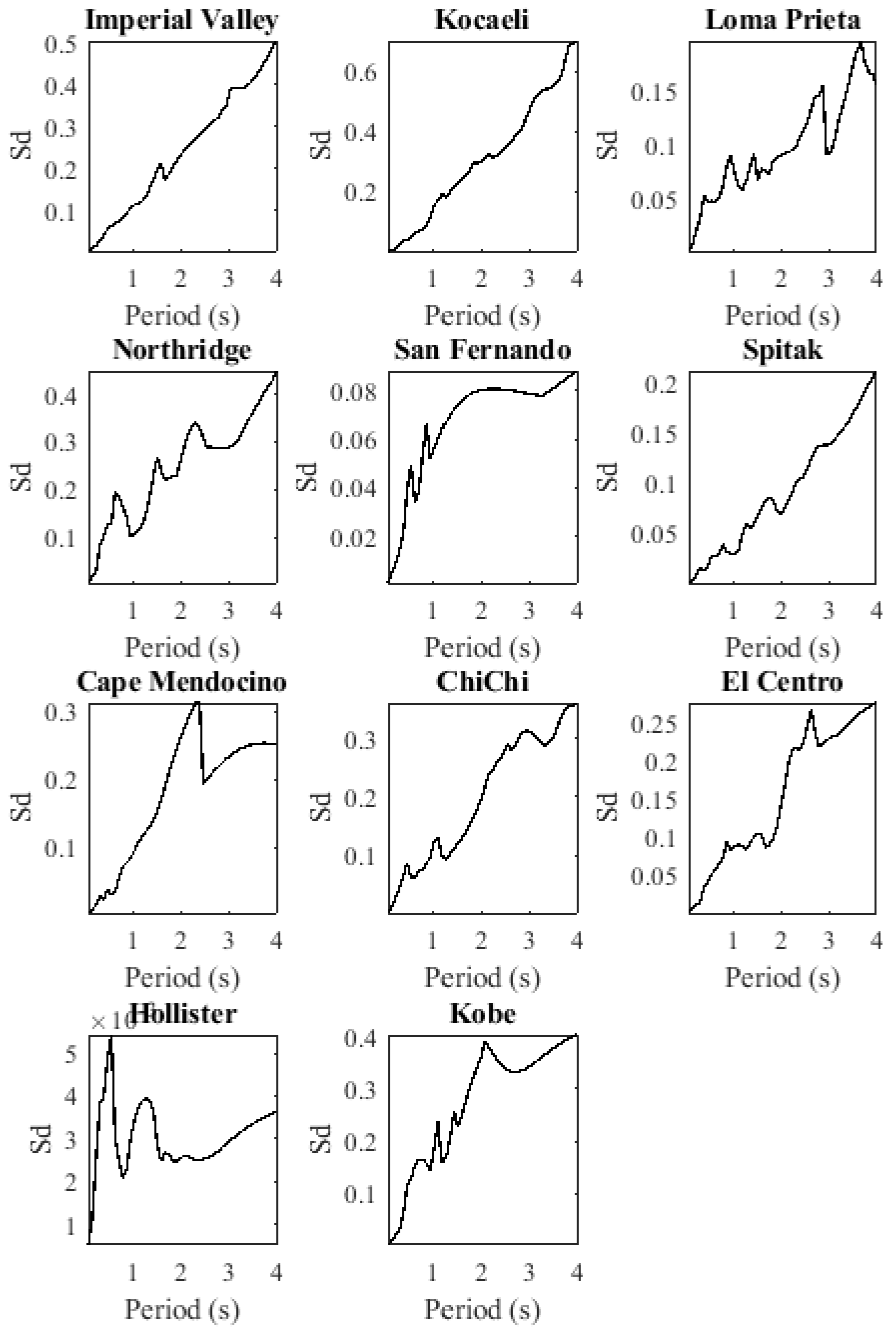
Initialize CDRS

```
CDRS=cell(numel(eqmotions),1);
% Calculation of peak values
for i=1:numel(eqmotions)
    % earthquake
    data=load([eqmotions{i},'.dat']);
    t=data(:,1);
    dt=t(2)-t(1);
    xgtt=data(:,2);
    S=OpenSeismoMatlab(dt,xgtt,sw,[],[],ksi,Tspectra,mu);
    CDRS{i}=[S.Period,S.CDSd,S.CDSv,S.CDPSa,S.fyK,S.muK,S.iterK];
end
```

## Output

Plot constant ductility spectral displacement

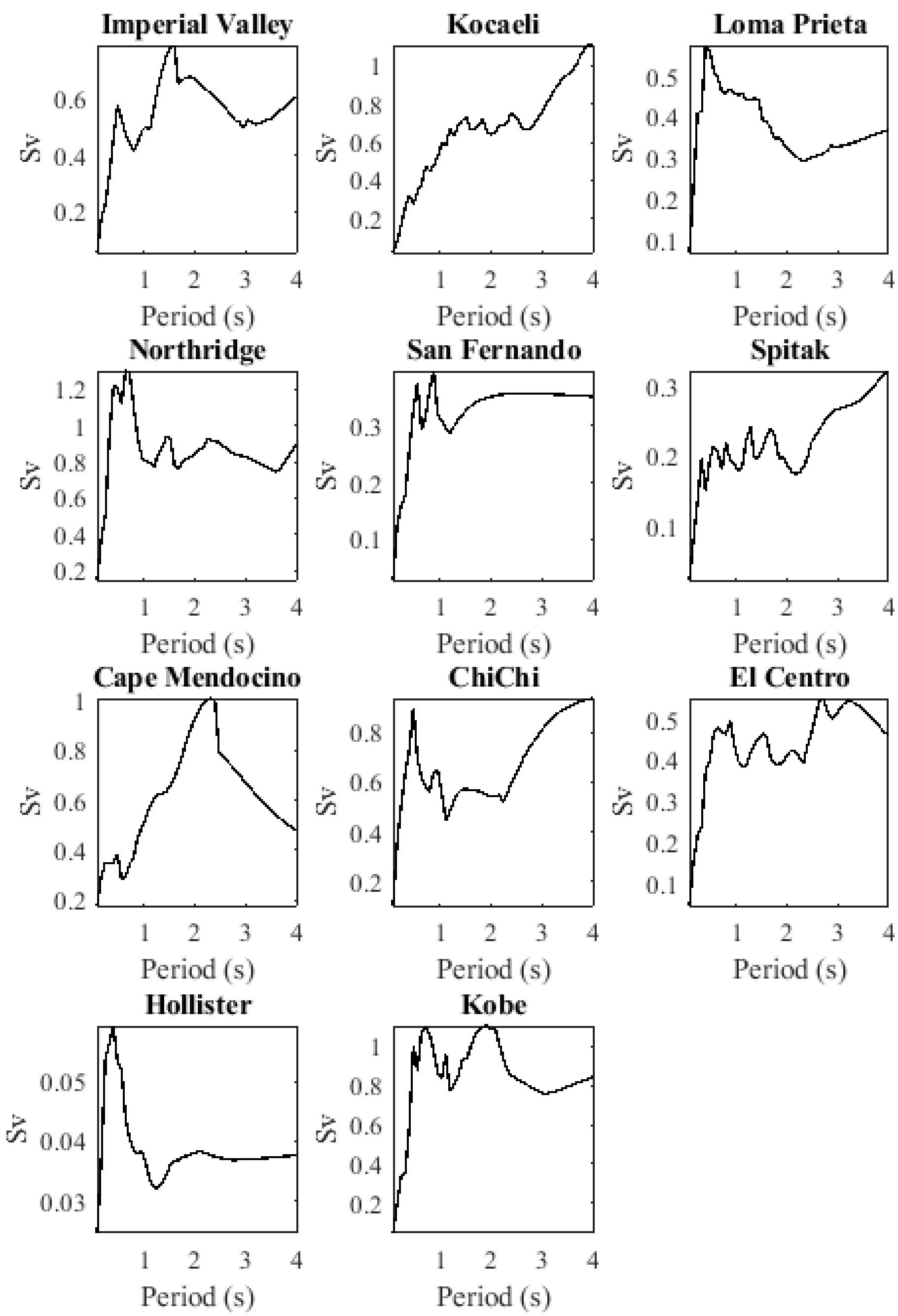
```
Fig1 = figure('units', 'centimeters', 'Position', [0,0, 20/sqrt(2), 20]);
% Scan all subplots
for i=1:numel(eqmotions)
    subplot(4,3,i)
    plot(CDRS{i}(:,1),CDRS{i}(:,2),'k','LineWidth',1);
    set(gca,'FontName','Times New Roman')
    title(eqmotions{i},'FontName','Times New Roman')
    ylabel('Sd','FontName','Times New Roman')
    xlabel('Period (s)','FontName','Times New Roman')
    axis tight
end
```




Plot constant ductility spectral velocity

```
Fig2 = figure('units', 'centimeters', 'Position', [0,0, 20/sqrt(2), 20]);
% Scan all subplots
for i=1:numel(eqmotions)
    subplot(4,3,i)
    plot(CDRS{i}(:,1),CDRS{i}(:,3),'k','LineWidth',1);
    set(gca,'FontName','Times New Roman')
    title(eqmotions{i},'FontName','Times New Roman')
    ylabel('Sv','FontName','Times New Roman')
    xlabel('Period (s)','FontName','Times New Roman')
```

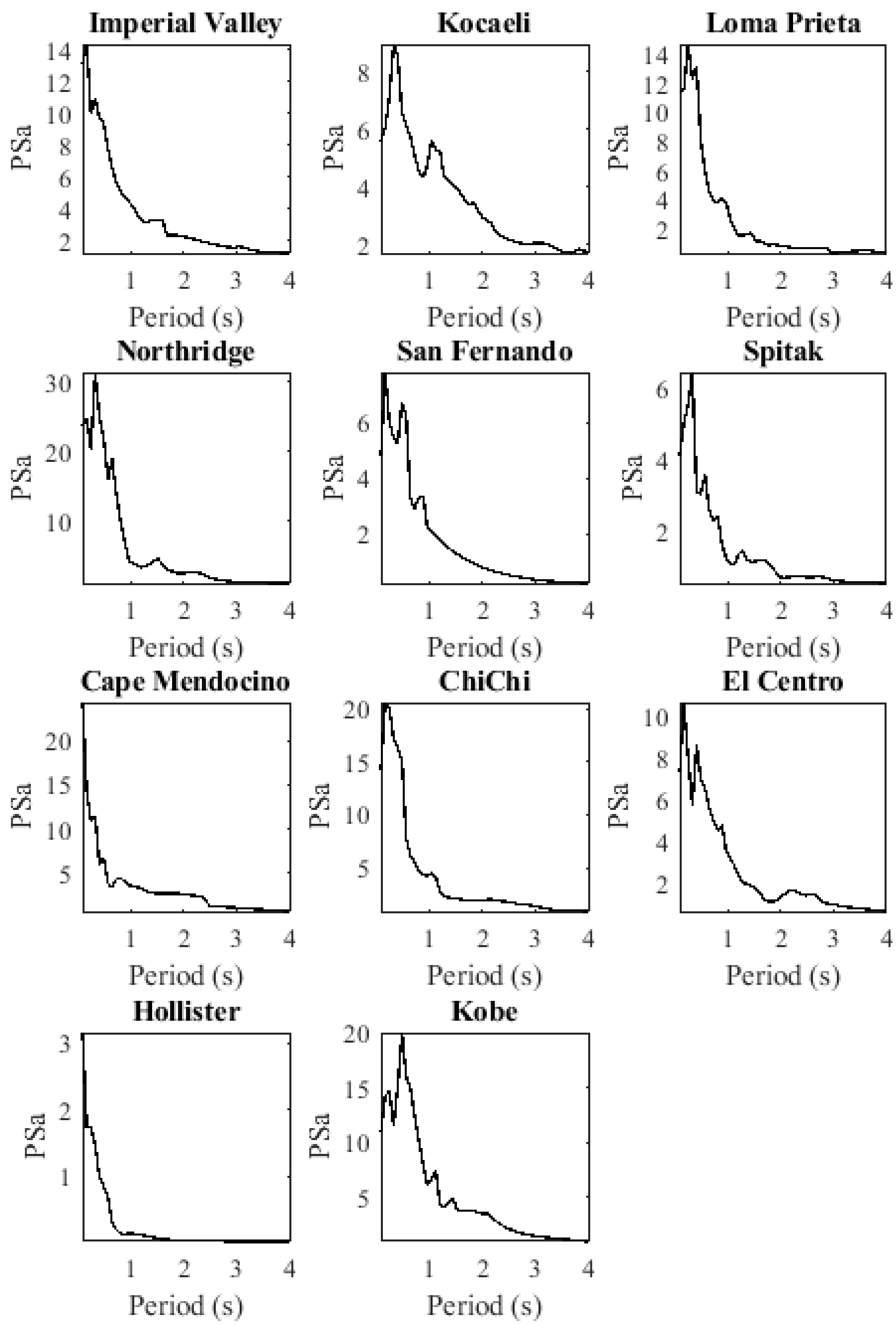
```
    axis tight
end
```




Plot constant ductility spectral acceleration

```
Fig3 = figure('units', 'centimeters', 'Position', [0,0, 20/sqrt(2), 20]);
% Scan all subplots
for i=1:numel(eqmotions)
    subplot(4,3,i)
    plot(CDRS{i}(:,1),CDRS{i}(:,4),'k','LineWidth',1);
```

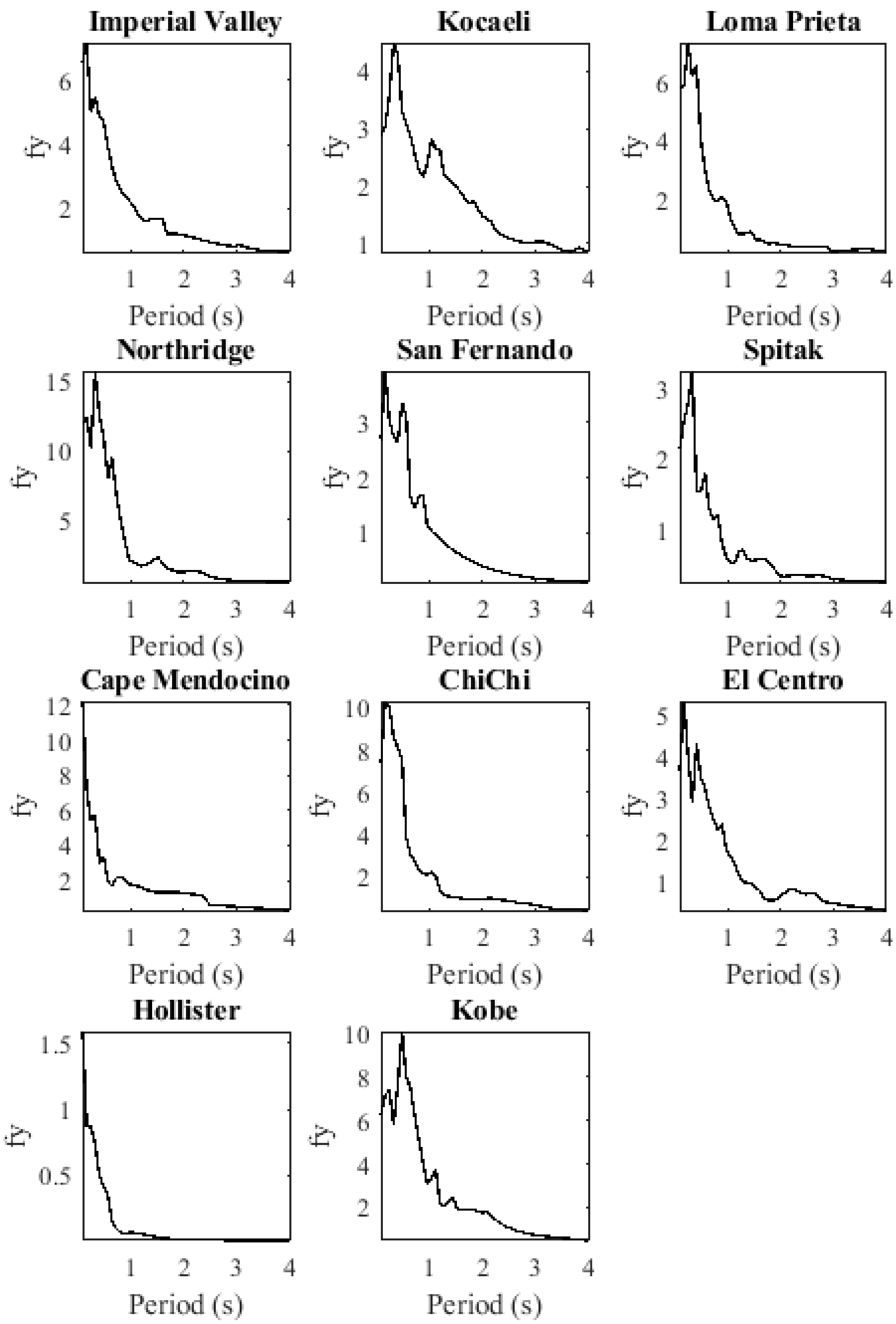
```
    set(gca,'FontName','Times New Roman')
    title(eqmotions{i},'FontName','Times New Roman')
    ylabel('PSa','FontName','Times New Roman')
    xlabel('Period (s)','FontName','Times New Roman')
    axis tight
end
```




Plot constant ductility spectral yield limit

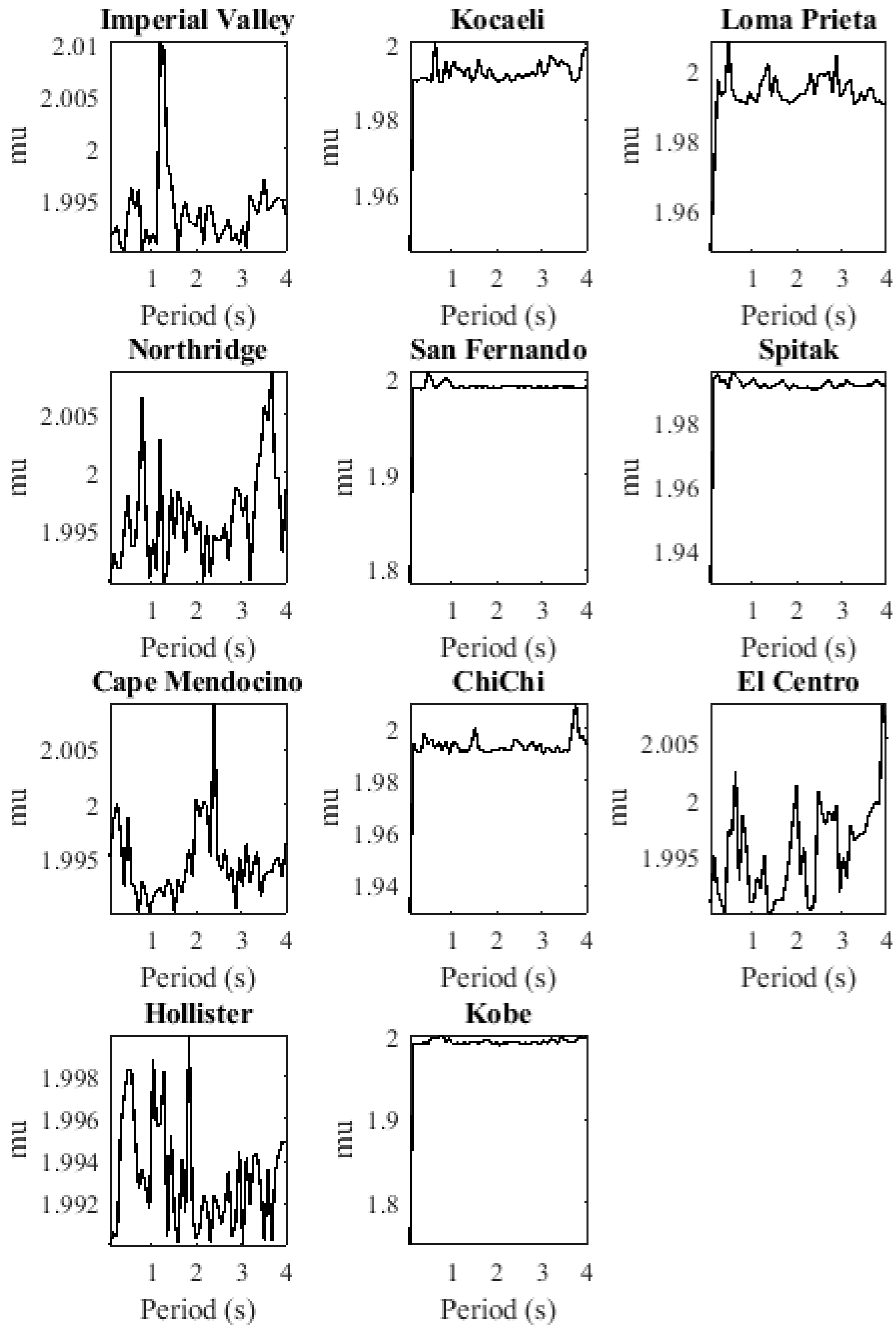
```
Fig4 = figure('units', 'centimeters', 'Position', [0,0, 20/sqrt(2), 20]);
```

```
% Scan all subplots
for i=1:numel(eqmotions)
    subplot(4,3,i)
    plot(CDRS{i}(:,1),CDRS{i}(:,5),'k','LineWidth',1);
    set(gca,'FontName','Times New Roman')
    title(eqmotions{i},'FontName','Times New Roman')
    ylabel('fy','FontName','Times New Roman')
    xlabel('Period (s)','FontName','Times New Roman')
    axis tight
end
```



Plot constant ductility spectral achieved ductility

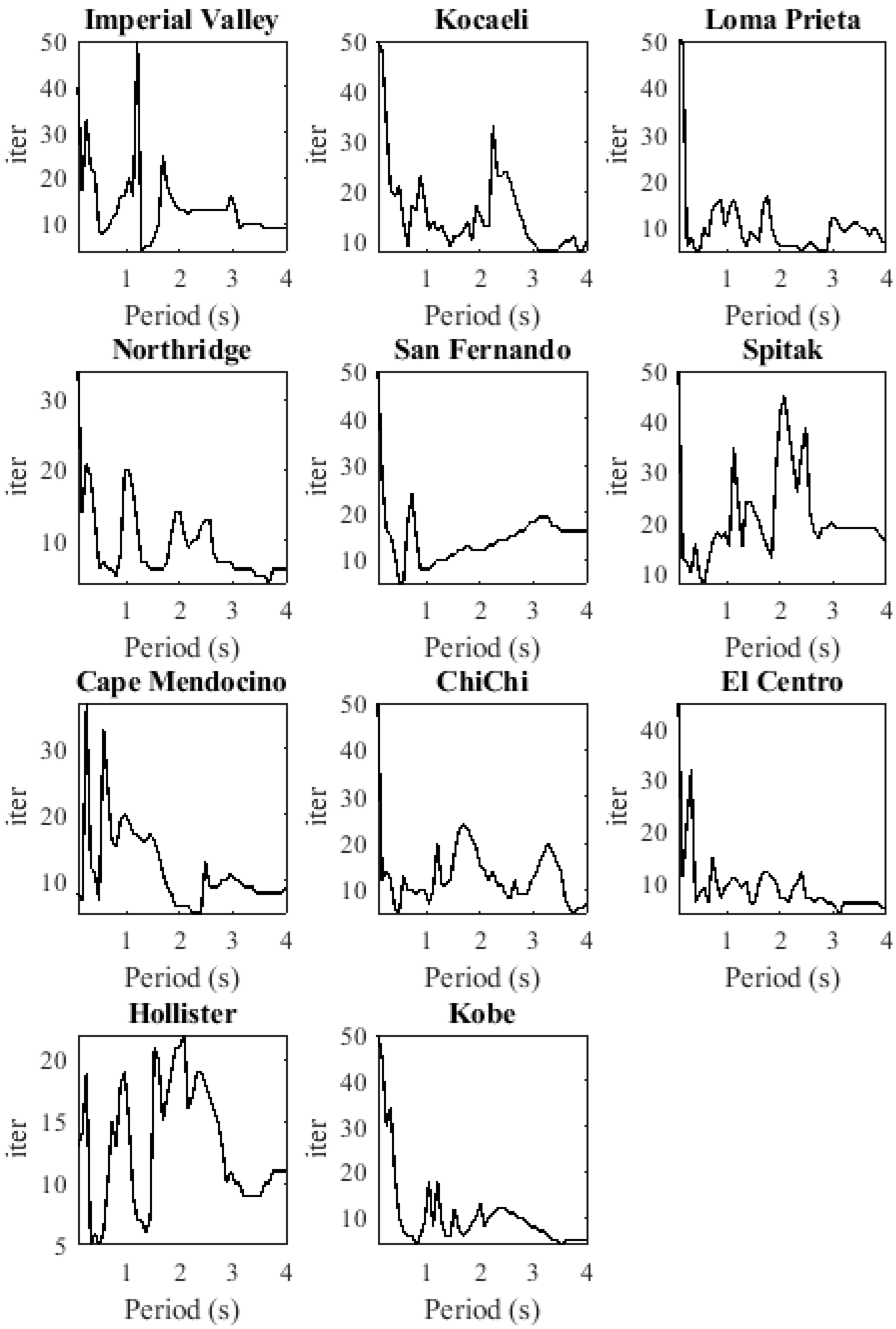
```
Fig5 = figure('units', 'centimeters', 'Position', [0,0, 20/sqrt(2), 20]);
% Scan all subplots
for i=1:numel(eqmotions)
    subplot(4,3,i)
    plot(CDRS{i}(:,1),CDRS{i}(:,6),'k','LineWidth',1);
    set(gca,'FontName','Times New Roman')
    title(eqmotions{i},'FontName','Times New Roman')
    ylabel('mu','FontName','Times New Roman')
    xlabel('Period (s)','FontName','Times New Roman')
    axis tight
end
```




Plot constant ductility spectral number of iterations needed for convergence

```
Fig6 = figure('units', 'centimeters', 'Position', [0,0, 20/sqrt(2), 20]);
% Scan all subplots
for i=1:numel(eqmotions)
    subplot(4,3,i)
    plot(CDRS{i}(:,1),CDRS{i}(:,7),'k','LineWidth',1);
    set(gca,'FontName','Times New Roman')
    title(eqmotions{i},'FontName','Times New Roman')
    ylabel('iter','FontName','Times New Roman')
    xlabel('Period (s)','FontName','Times New Roman')
```

```
    axis tight
end
```




## Copyright




- Major, Infrastructure Engineer, Hellenic Air Force
- Civil Engineer, M.Sc., Ph.D.
- Email: gpapazafeiropoulos@yahoo.gr

# Fourier spectra

Generate the Fourier spectra of an earthquake suite using OpenSeismoMatlab.

## Contents



## Input

earthquake motions

```
eqmotions={'Imperial Valley'; % Imperial valley 1979
    'Kocaeli';
    'Loma Prieta';
    'Northridge';
    'San Fernando';
    'Spitak';
    'Cape Mendocino';
    'ChiChi';
    'El Centro'; % Imperial valley 1940
    'Hollister';
    'Kobe'};
```

Set the eigenperiod range for which the response spectra will be calculated.

```
Tspectra=(0.02:0.01:4)';
```

Set critical damping ratio of the response spectra to be calculated.

```
ksi=0.05;
```

Set the target ductility (not used here)

```
mu=2;
```

Extract fourier spectra

```
sw='fas';
```

## Calculation

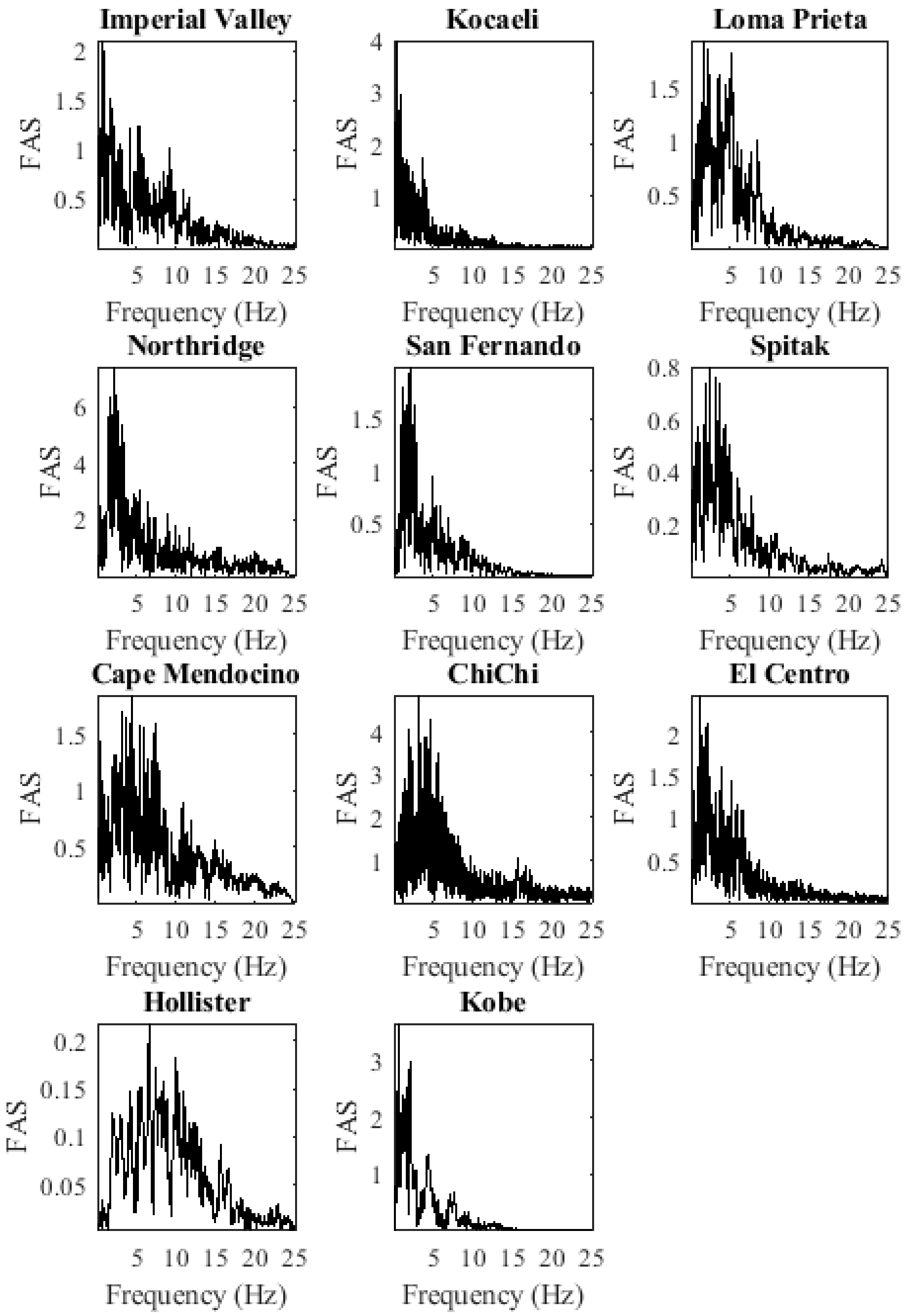
Initialize Fourier

```
Fourier=cell(numel(eqmotions),1);
% Calculation of peak values
for i=1:numel(eqmotions)
    % earthquake
    data=load([eqmotions{i},'.dat']);
    t=data(:,1);
    dt=t(2)-t(1);
    xgtt=data(:,2);
    S=OpenSeismoMatlab(dt,xgtt,sw);
    Fourier{i}=[S.freq,S.FAS];
end
```

## Output

Plot Fourier amplitude

```
Fig1 = figure('units', 'centimeters', 'Position', [0,0, 20/sqrt(2), 20]);
% Scan all subplots
for i=1:numel(eqmotions)
    subplot(4,3,i)
    plot(Fourier{i}(:,1),Fourier{i}(:,2),'k','LineWidth',1);
    set(gca,'FontName','Times New Roman')
    title(eqmotions{i},'FontName','Times New Roman')
    ylabel('FAS','FontName','Times New Roman')
    xlabel('Frequency (Hz)','FontName','Times New Roman')
    axis tight
end
```




## Copyright




- Major, Infrastructure Engineer, Hellenic Air Force
- Civil Engineer, M.Sc., Ph.D.
- Email: gpapazafeiropoulos@yahoo.gr

# Linear elastic response spectra

Generate the linear elastic response spectra of an earthquake suite using OpenSeismoMatlab.

## Contents



## Input

Earthquake motions

```
eqmotions={'Imperial Valley'; % Imperial valley 1979
    'Kocaeli';
    'Loma Prieta';
    'Northridge';
    'San Fernando';
    'Spitak';
    'Cape Mendocino';
    'ChiChi';
    'El Centro'; % Imperial valley 1940
    'Hollister';
    'Kobe'};
```

Set the eigenperiod range for which the response spectra will be calculated.

```
Tspectra=(0.01:0.01:4)';
```

Set critical damping ratio of the response spectra to be calculated.

```
ksi=0.05;
```

Set the target ductility (not used here)

```
mu=2;
```

Extract linear elastic response spectra

```
sw='es';
```

## Calculation

Initialize LERS

```
LERS=cell(numel(eqmotions),1);
```

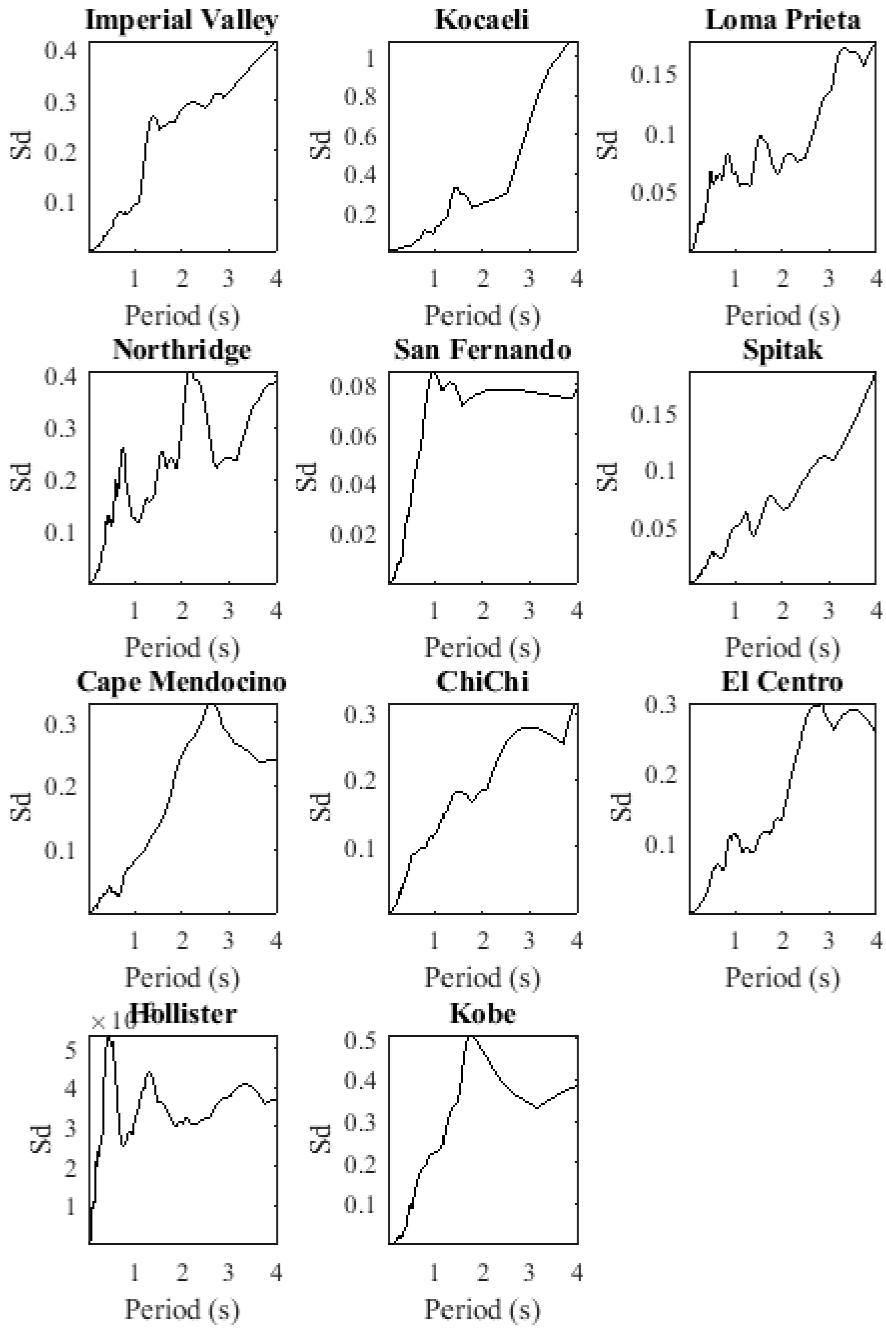
Calculation of peak values

```
for i=1:numel(eqmotions)
    % earthquake
    data=load([eqmotions{i},'.dat']);
    t=data(:,1);
    dt=t(2)-t(1);
    xgtt=data(:,2);
    S=OpenSeismoMatlab(dt,xgtt,sw,[],[],ksi,Tspectra,mu);
    LERS{i}=[S.Period,S.Sd,S.PSv,S.PSa];
end
```

## Output

Plot displacement response spectra

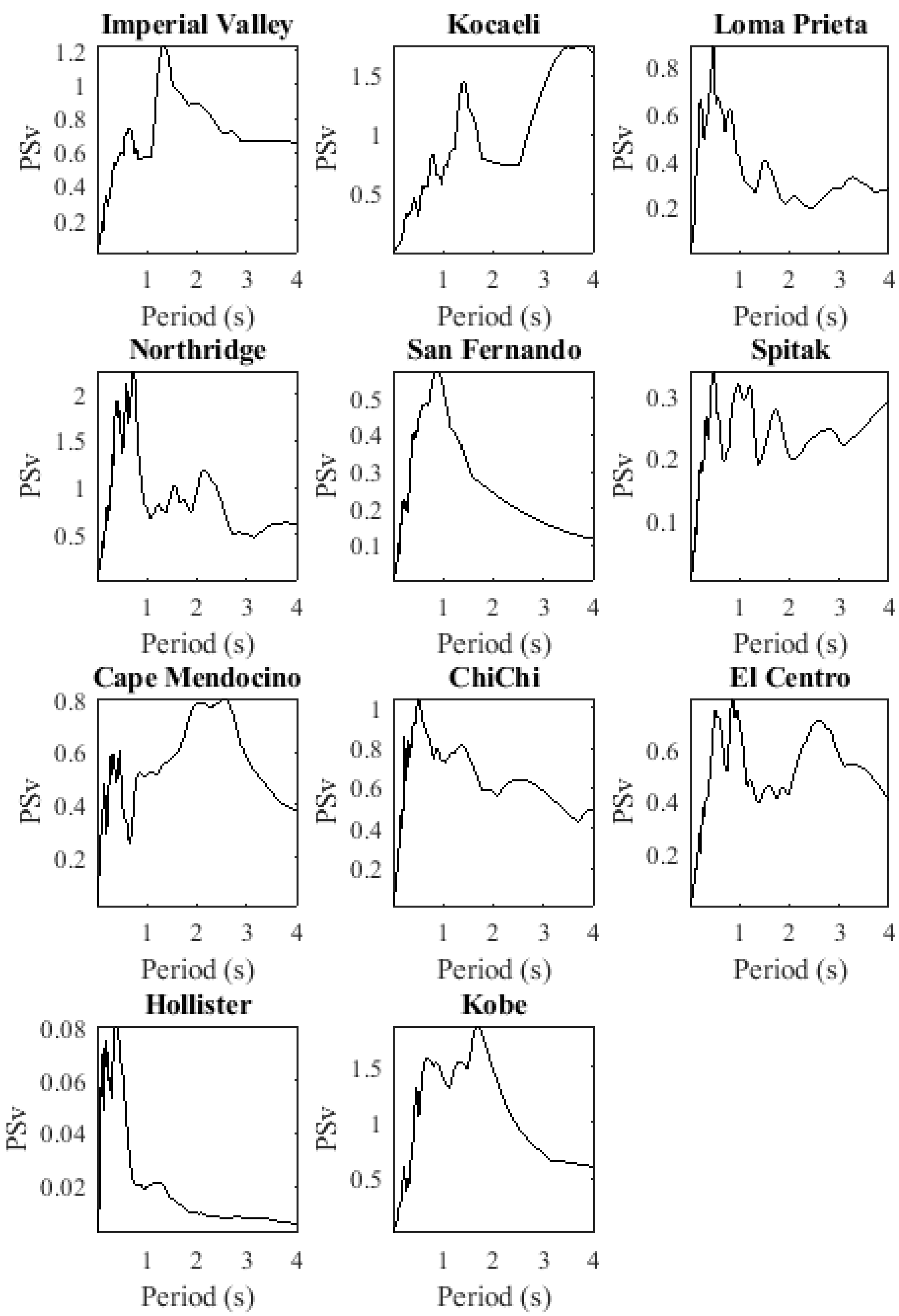
```
Fig1 = figure('units', 'centimeters', 'Position', [0,0, 20/sqrt(2), 20]);
% Scan all subplots
for i=1:numel(eqmotions)
    subplot(4,3,i)
    plot(LERS{i}(:,1),LERS{i}(:,2),'k','LineWidth',1);
    set(gca,'FontName','Times New Roman')
    title(eqmotions{i},'FontName','Times New Roman')
    ylabel('Sd','FontName','Times New Roman')
    xlabel('Period (s)','FontName','Times New Roman')
    axis tight
end
```




Plot pseudo-velocity response spectra

```
Fig2 = figure('units', 'centimeters', 'Position', [0,0, 20/sqrt(2), 20]);
% Scan all subplots
for i=1:numel(eqmotions)
    subplot(4,3,i)
    plot(LERS{i}(:,1),LERS{i}(:,3),'k','LineWidth',1);
    set(gca,'FontName','Times New Roman')
    title(eqmotions{i},'FontName','Times New Roman')
    ylabel('PSv','FontName','Times New Roman')
    xlabel('Period (s)','FontName','Times New Roman')
```

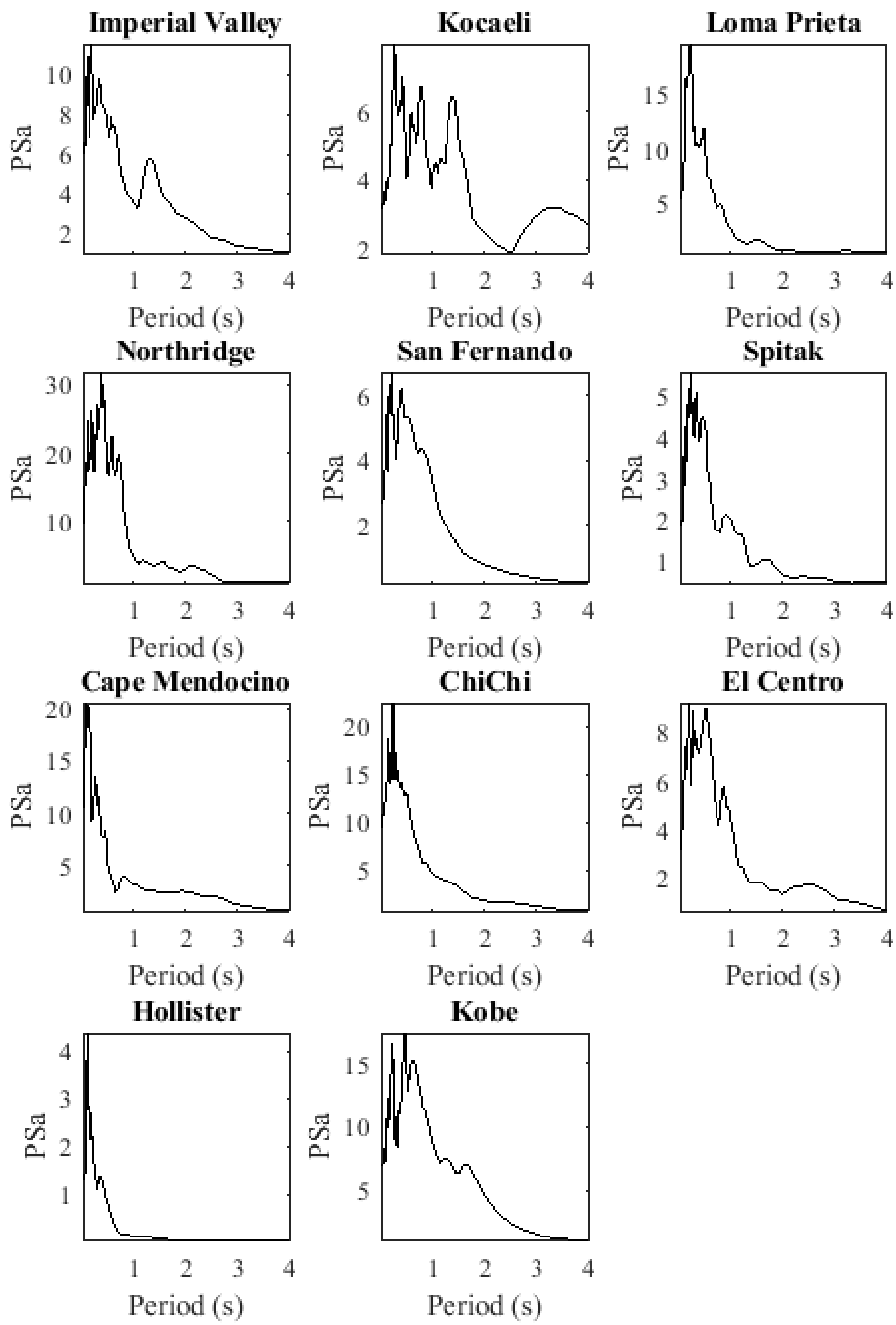
```
    axis tight
end
```




Plot pseudo-acceleration response spectra

```
Fig3 = figure('units', 'centimeters', 'Position', [0,0, 20/sqrt(2), 20]);
% Scan all subplots
for i=1:numel(eqmotions)
    subplot(4,3,i)
    plot(LERS{i}(:,1),LERS{i}(:,4),'k','LineWidth',1);
```

```
    set(gca,'FontName','Times New Roman')
    title(eqmotions{i},'FontName','Times New Roman')
    ylabel('PSa','FontName','Times New Roman')
    xlabel('Period (s)','FontName','Times New Roman')
    axis tight
end
```




## Copyright

- Major, Infrastructure Engineer, Hellenic Air Force
- Civil Engineer, M.Sc., Ph.D.
- Email: gpapazafeiropoulos@yahoo.gr

---

*Published with MATLAB® R2017b*

# Verification

This is to verify that OpenSeismoMatlab works properly for all of its possible options and types of application

## Contents



## Load earthquake data

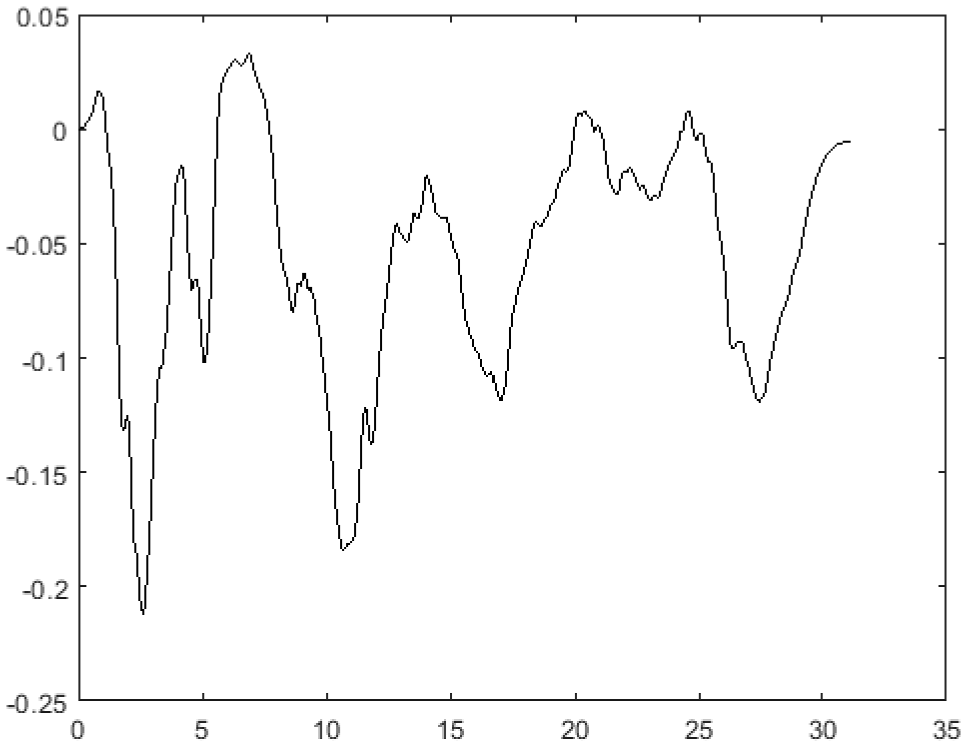
Earthquake acceleration time history of the El Centro earthquake will be used (El Centro, 1940, El Centro Terminal Substation Building)

```
fid=fopen('elcentro.dat','r');
text=textscan(fid,'%f %f');
fclose(fid);
t=text{1,1};
dt=t(2)-t(1);
xgtt=9.81*text{1,2};
```

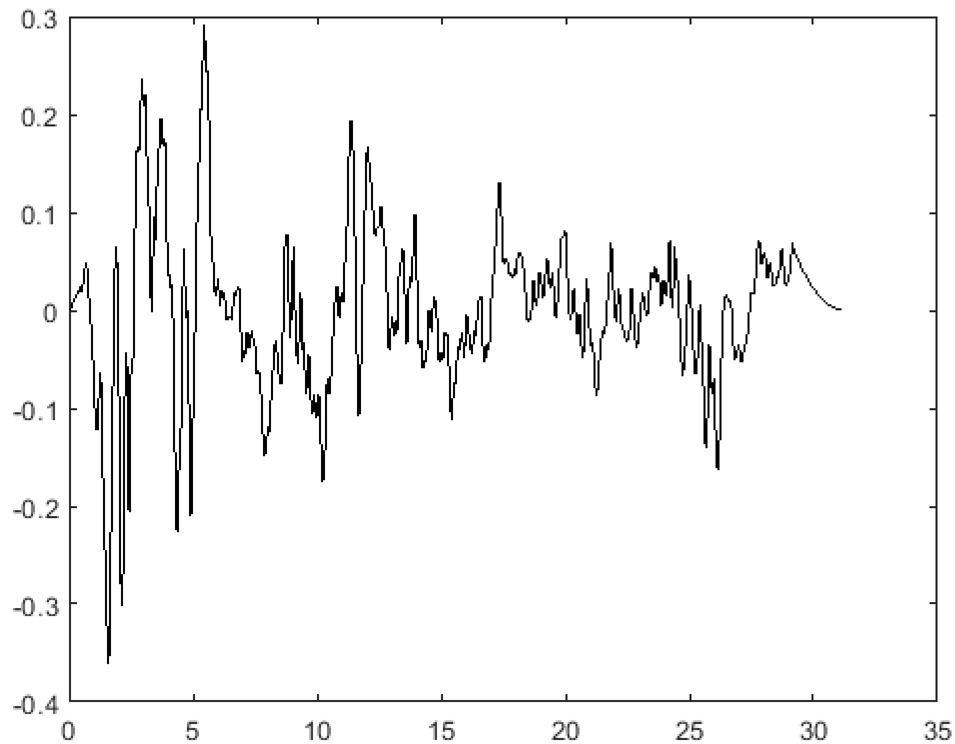
## Time histories without baseline correction

```
sw='timehist';
baselineSw=false;
S1=OpenSeismoMatlab(dt,xgtt,sw,baselineSw);
```

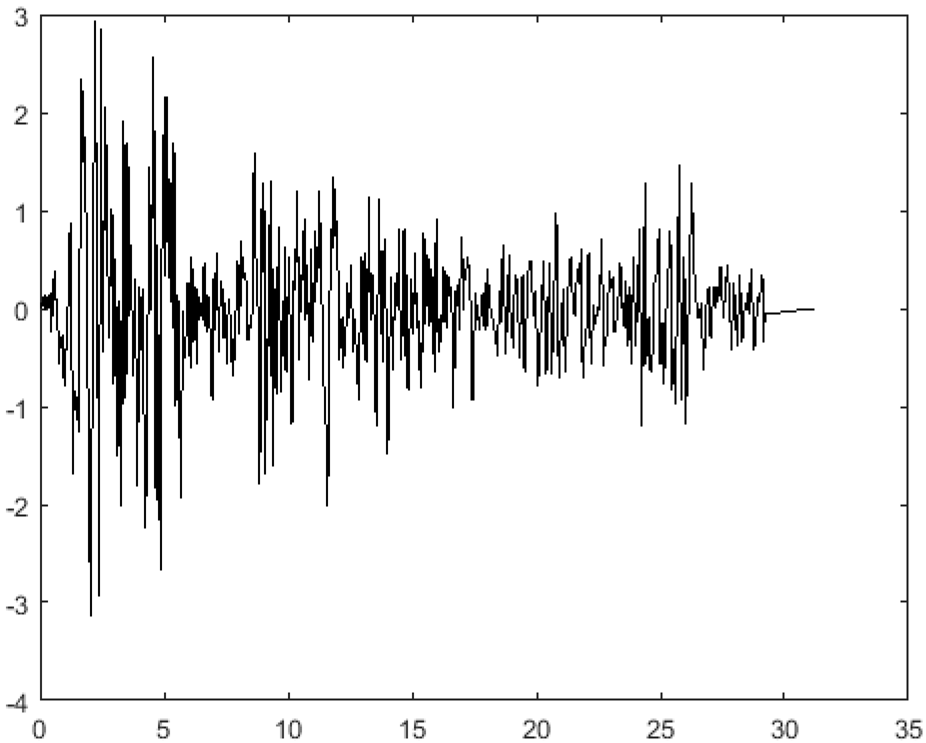
```
figure(1)
plot(S1.time,S1.disp,'k','LineWidth',1)
```




```
figure(2)
plot(S1.time,S1.vel,'k','LineWidth',1)
```



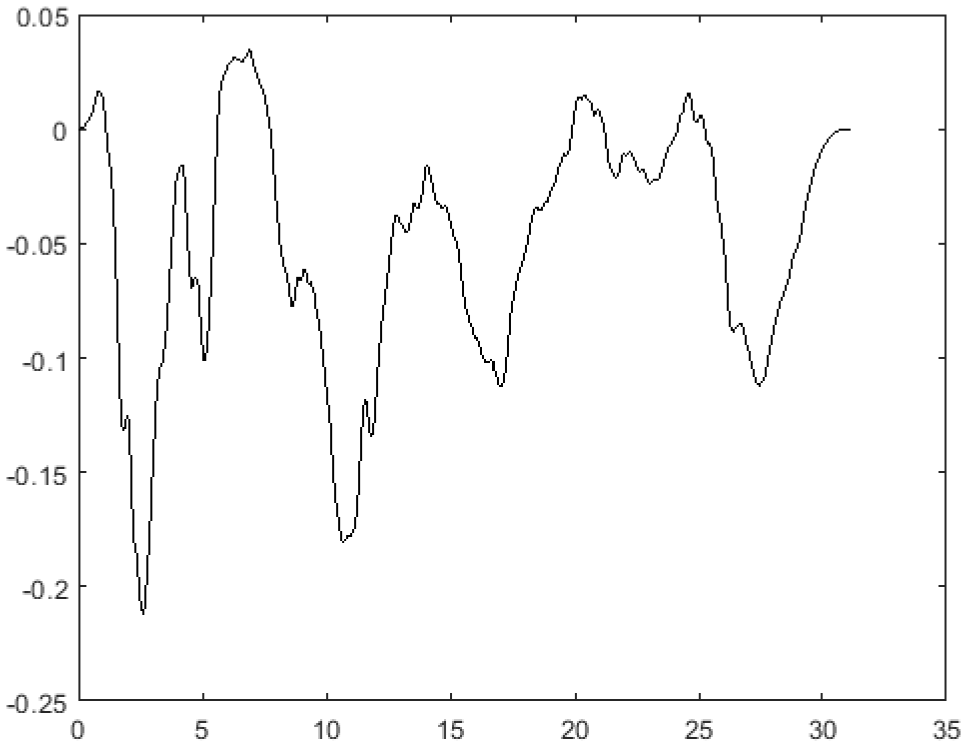
```
figure(3)
plot(S1.time,S1.acc,'k','LineWidth',1)
```



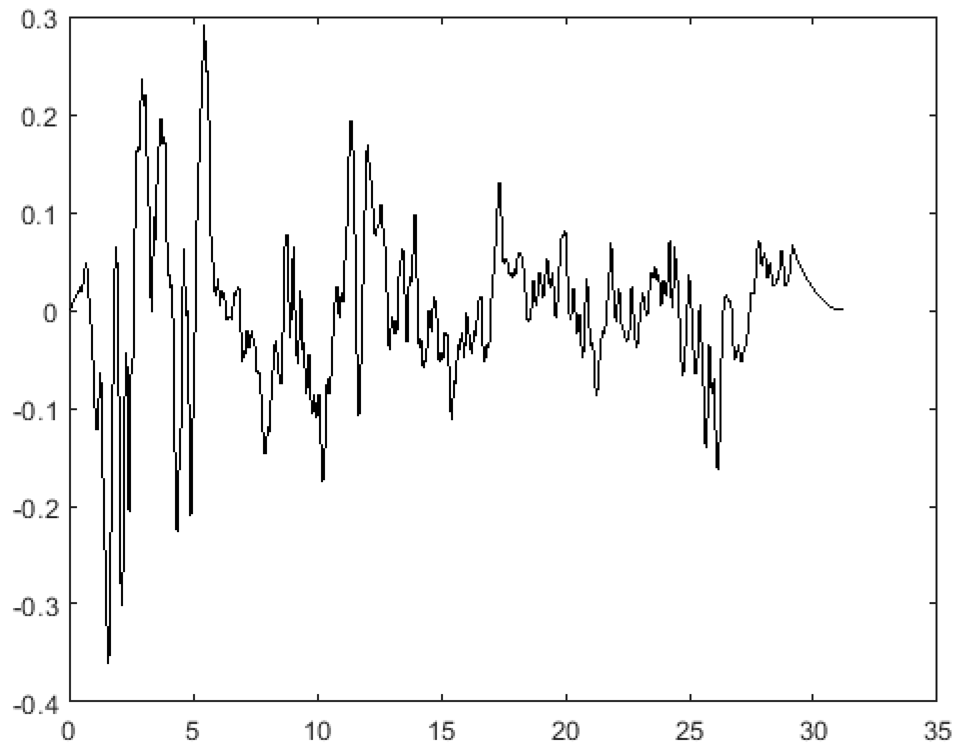

### Time histories with baseline correction

```
sw='timehist';
baselineSw=true;
S2=OpenSeismoMatlab(dt,xgtt,sw,baselineSw);
```

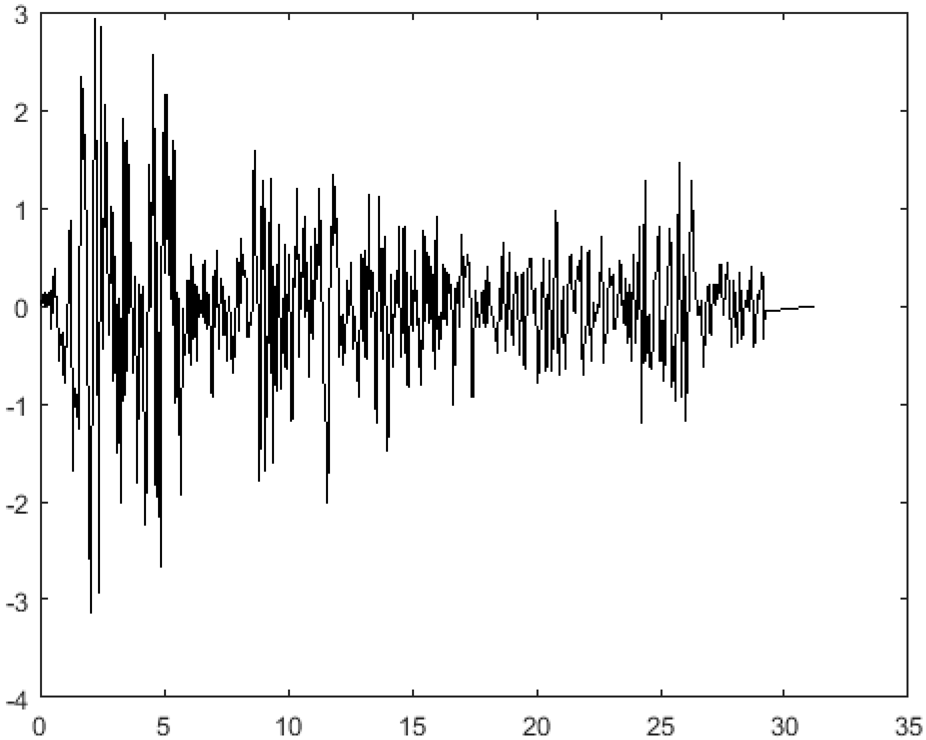
```
figure(4)
plot(S2.time,S2.disp,'k','LineWidth',1)
```




```
figure(5)
plot(S2.time,S2.vel,'k','LineWidth',1)
```



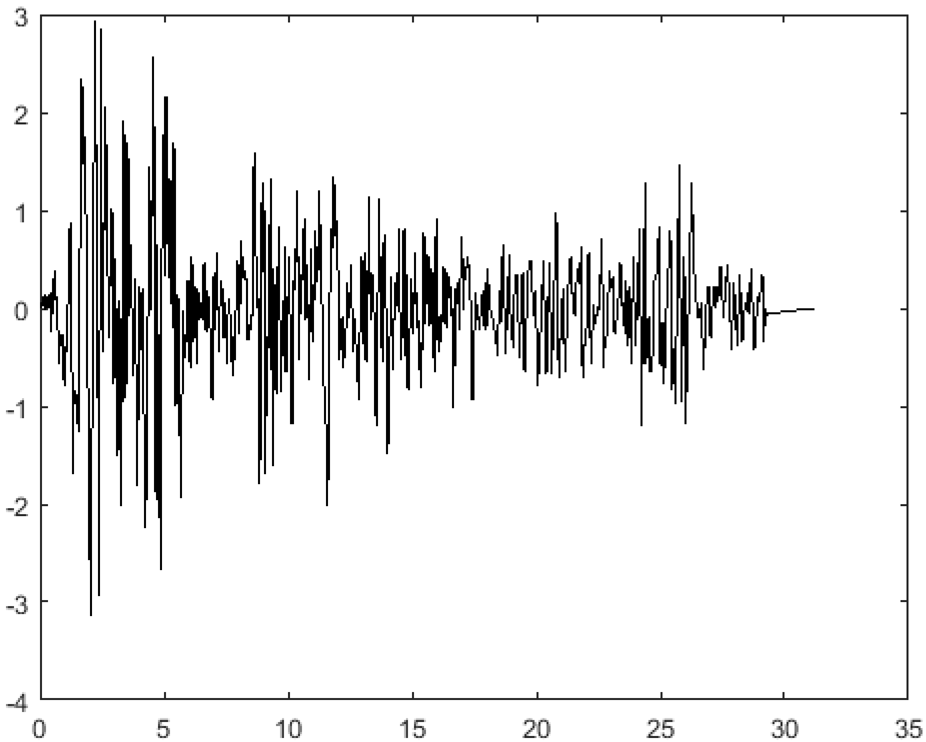
```
figure(6)
plot(S2.time,S2.acc,'k','LineWidth',1)
```




### Resample acceleration time history from 0.02 sec to 0.01 sec.

```
sw='resample';
dti=0.01;
S3=OpenSeismoMatlab(dt,xgtt,sw,[],dti);
```

```
figure(7)
plot(S3.time,S3.acc,'k','LineWidth',1)
```




### PGA

```
sw='pga';
S4=OpenSeismoMatlab(dt,xgtt,sw);
```

```
S4.PGA
```

```
ans =

   3.127624200000000
```

### PGV

```
sw='pgv';
S5=OpenSeismoMatlab(dt,xgtt,sw);
```

```
S5.PGV
```

```
ans =

   0.360920691000000
```

## PGD

```
sw='pgd';
S6=OpenSeismoMatlab(dt,xgtt,sw);
```

```
S6.PGD
```

```
ans =

   0.211893410160001
```

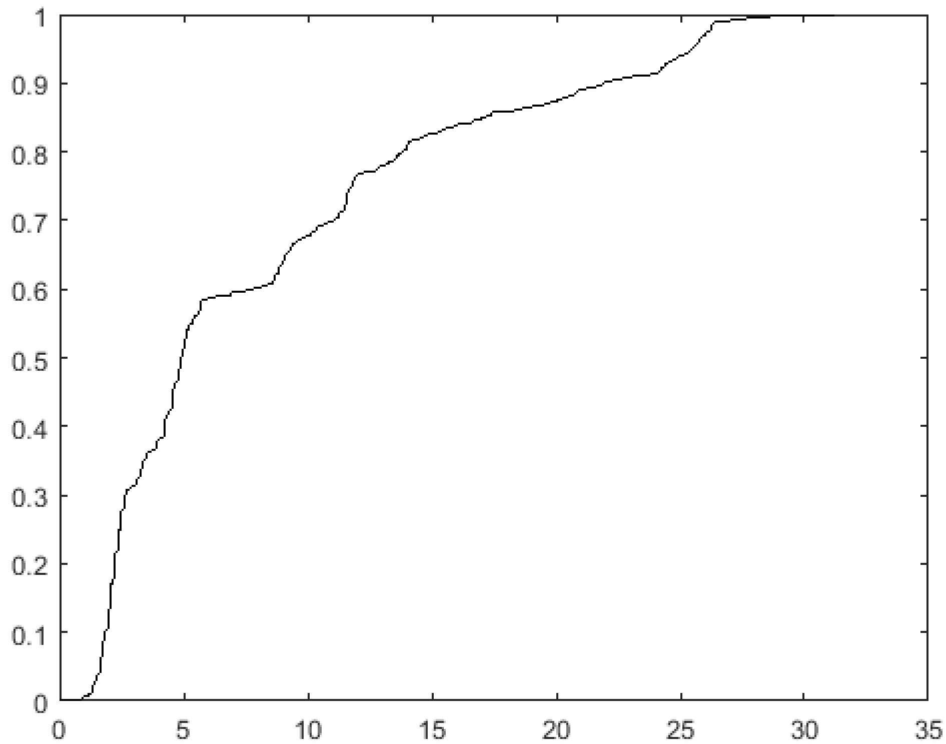
## Arias intensity and significant duration

```
sw='arias';
S7=OpenSeismoMatlab(dt,xgtt,sw);
```

```
S7.Ecum
```

```
ans =

  11.251388628141965
```

```
figure(8)
plot(S7.time,S7.EcumTH,'k','LineWidth',1)
```




```
S7.t_5_95
```

```
ans =

   1.680000000000000  25.500000000000000
```

```
S7.Td
```

```
ans =

  23.840000000000000
```

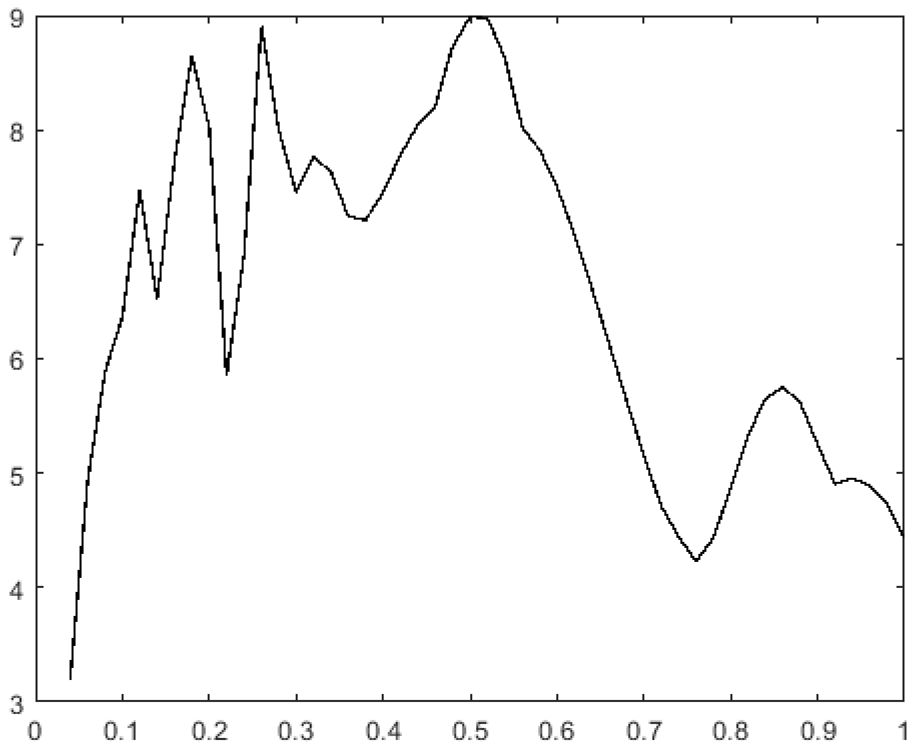
```
S7.arias
```

```
ans =

   1.645606908074047
```

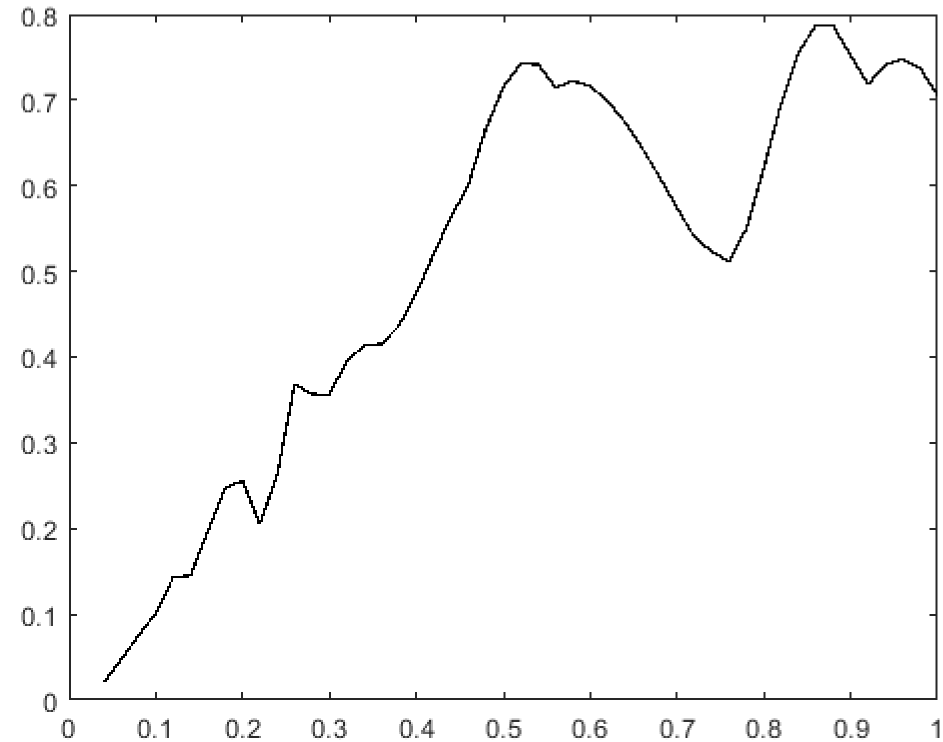
## Linear elastic response spectra and pseudospectra

```
sw='es';
ksi=0.05;
T=0.04:0.02:1;
S8=OpenSeismoMatlab(dt,xgtt,sw,[],[],ksi,T);
```

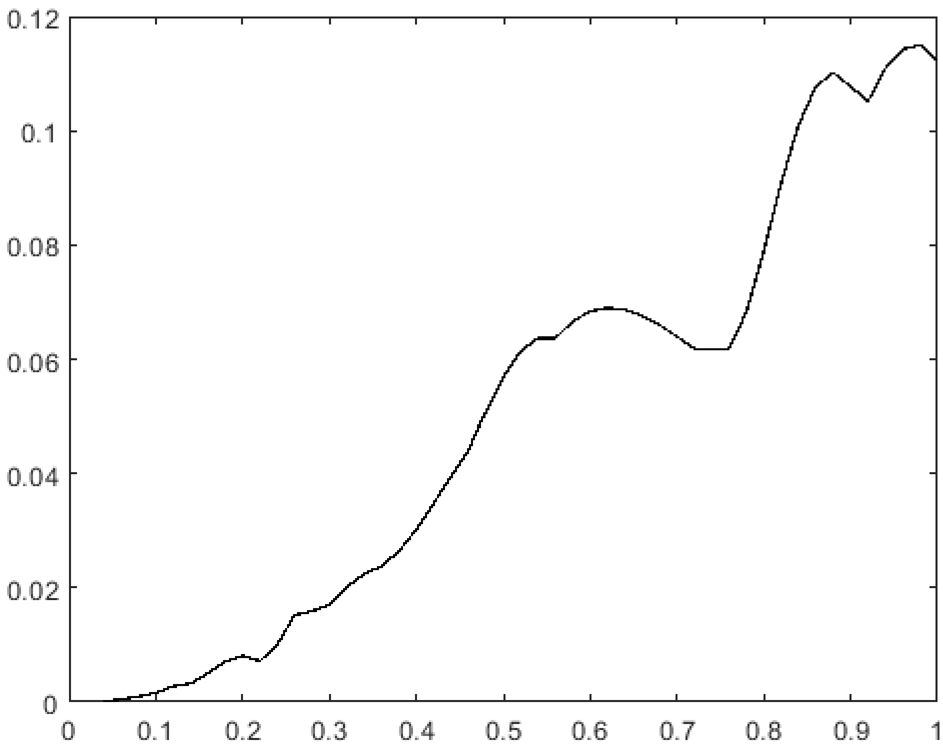
```
figure(9)
plot(S8.Period,S8.PSa,'k','LineWidth',1)
```



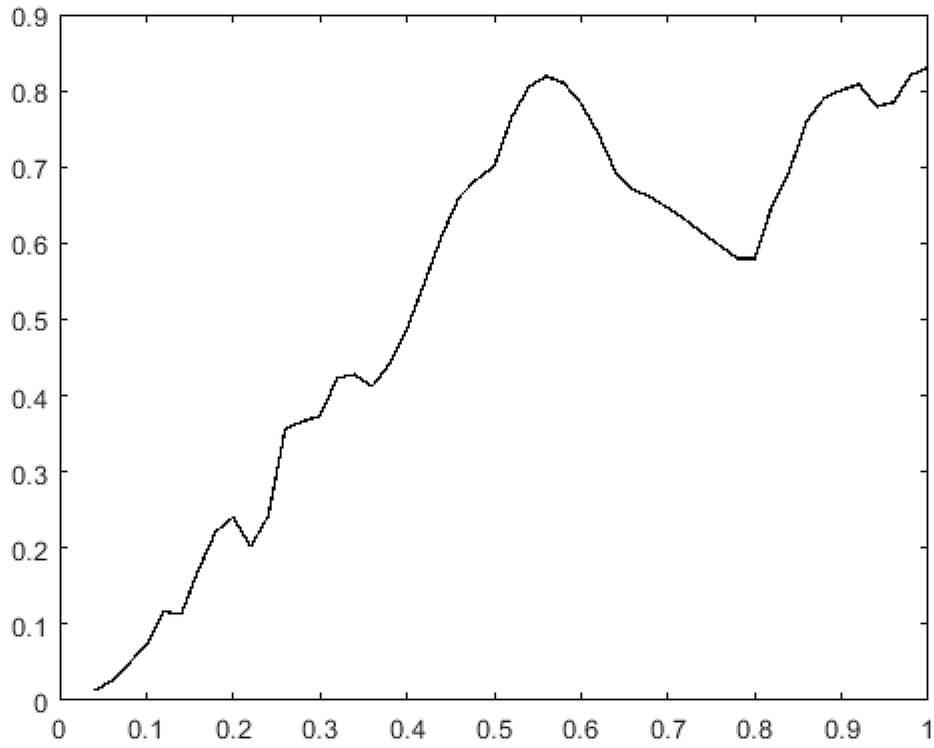
```
figure(10)
plot(S8.Period,S8.PSv,'k','LineWidth',1)
```



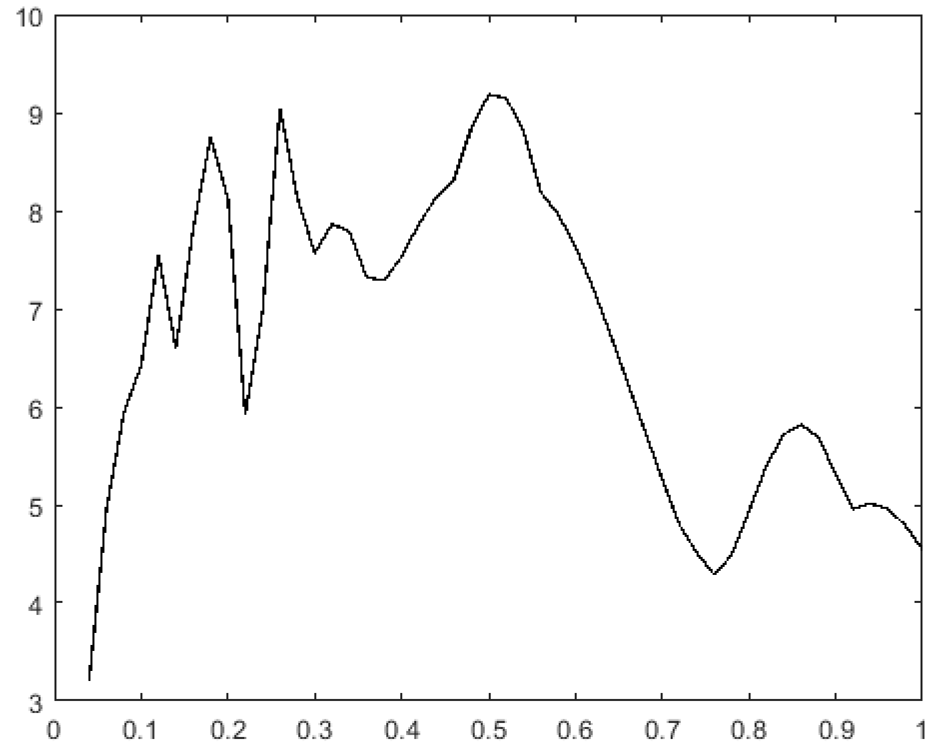

```
figure(11)
plot(S8.Period,S8.Sd,'k','LineWidth',1)
```



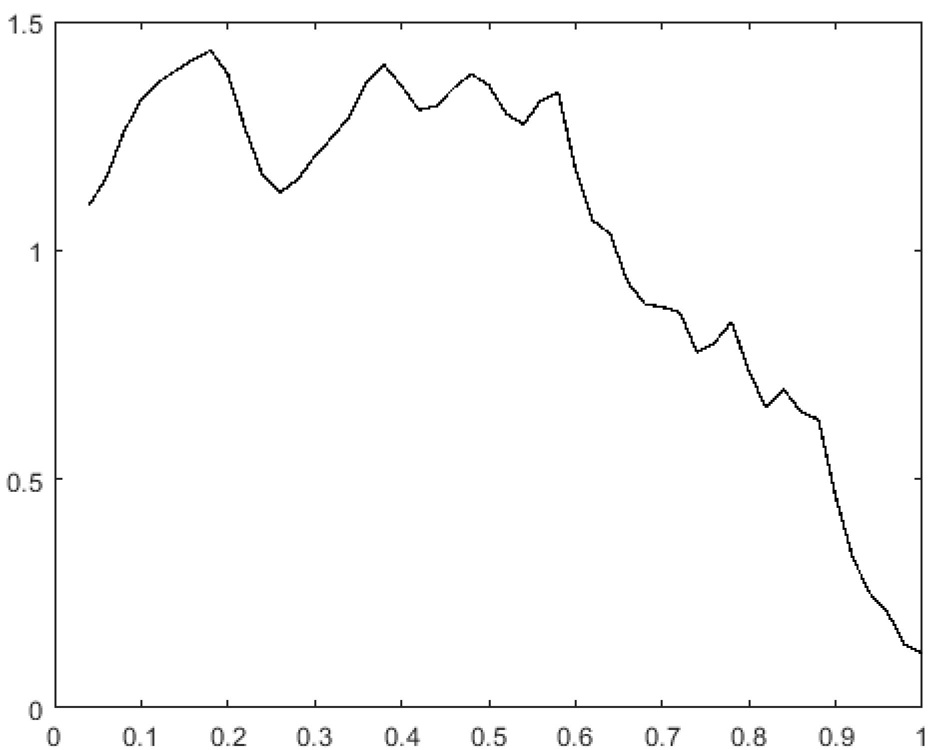
```
figure(12)
plot(S8.Period,S8.Sv,'k','LineWidth',1)
```




```
figure(13)
plot(S8.Period,S8.Sa,'k','LineWidth',1)
```

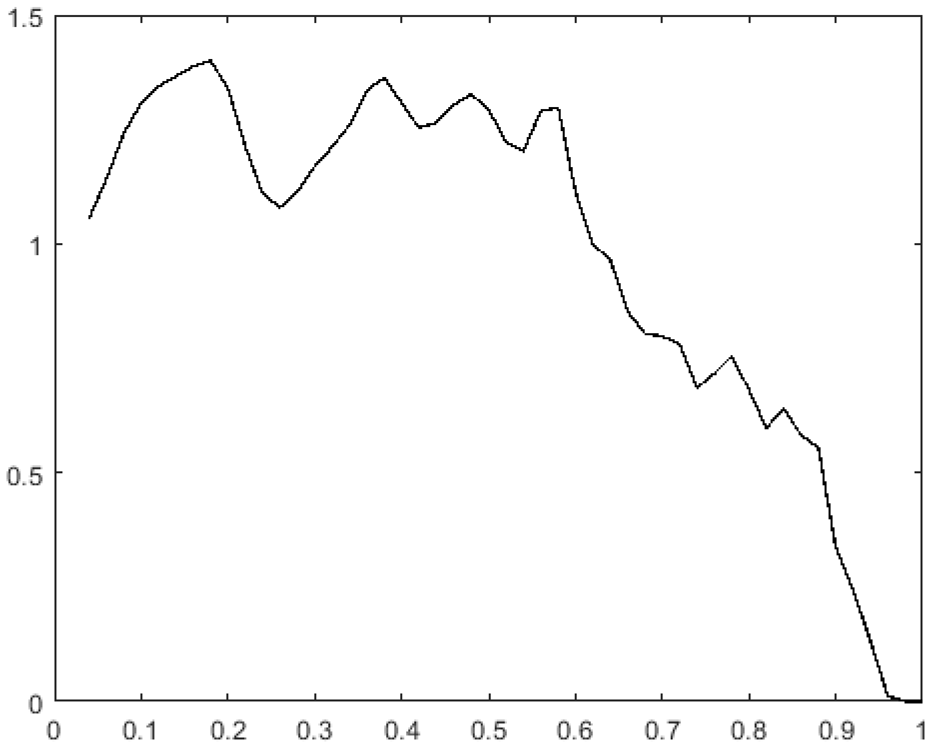



```
figure(14)
plot(S8.Period,S8.SievABS,'k','LineWidth',1)
```



```
figure(15)
plot(S8.Period,S8.SievREL,'k','LineWidth',1)
```

```
Warning: Imaginary parts of complex X
and/or Y arguments ignored
```




```
S8.PredPSa
```

```
ans =

   8.990993548805100
```

```
S8.PredPeriod
```

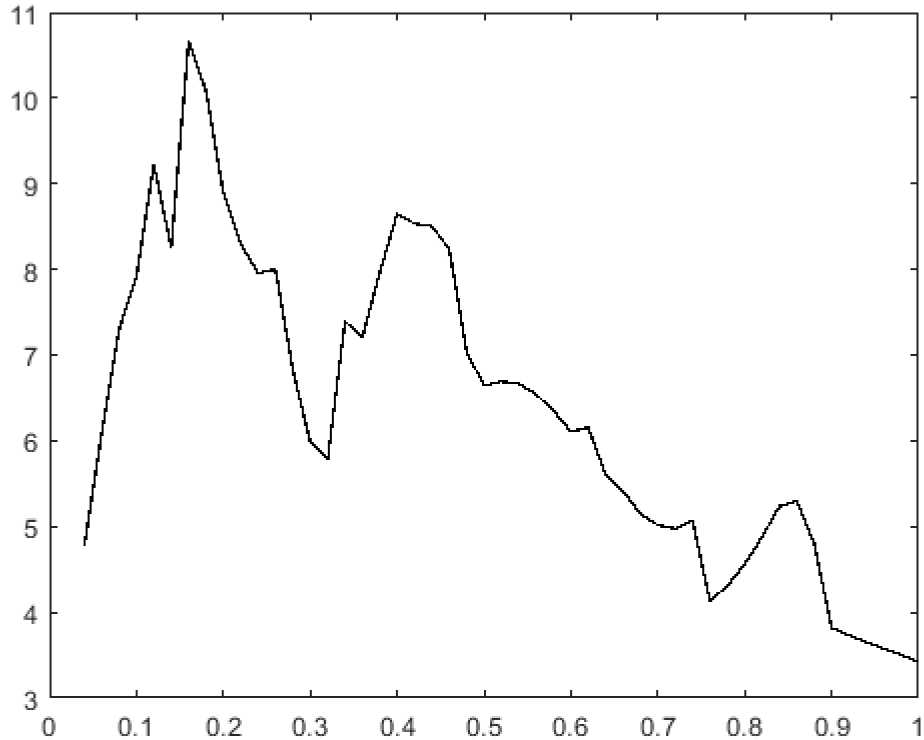
```
ans =

   0.500000000000000
```

## Constant ductility response spectra and pseudospectra

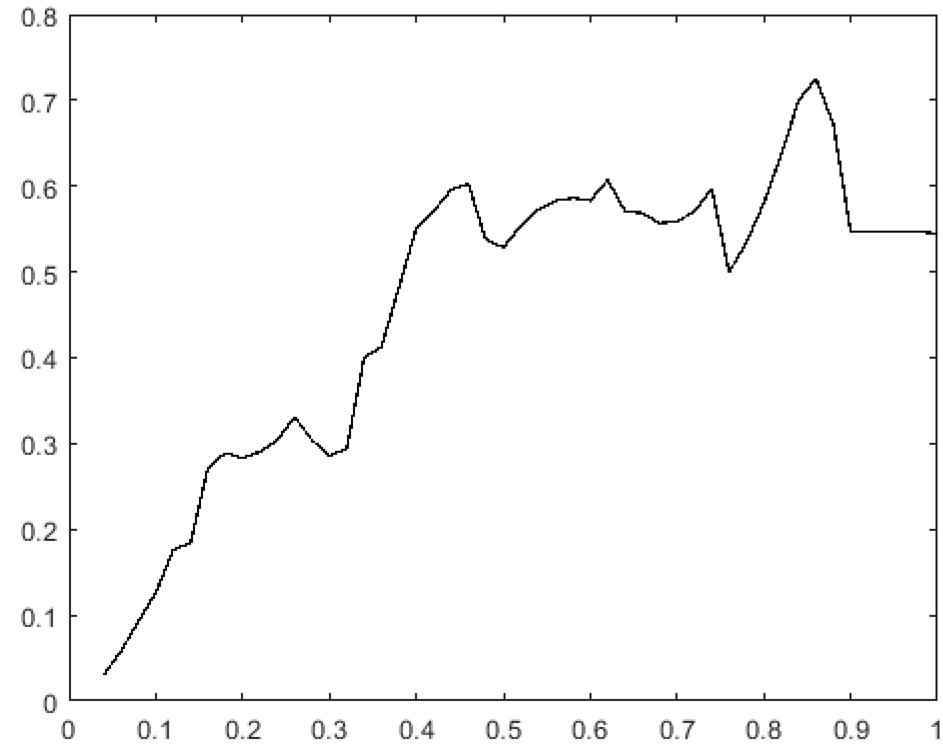
```
sw='cds';
ksi=0.05;
```

```
T=0.04:0.02:1;
mu=2;
S9=OpenSeismoMatlab(dt,xgtt,sw,[],[],ksi,T,mu);
```

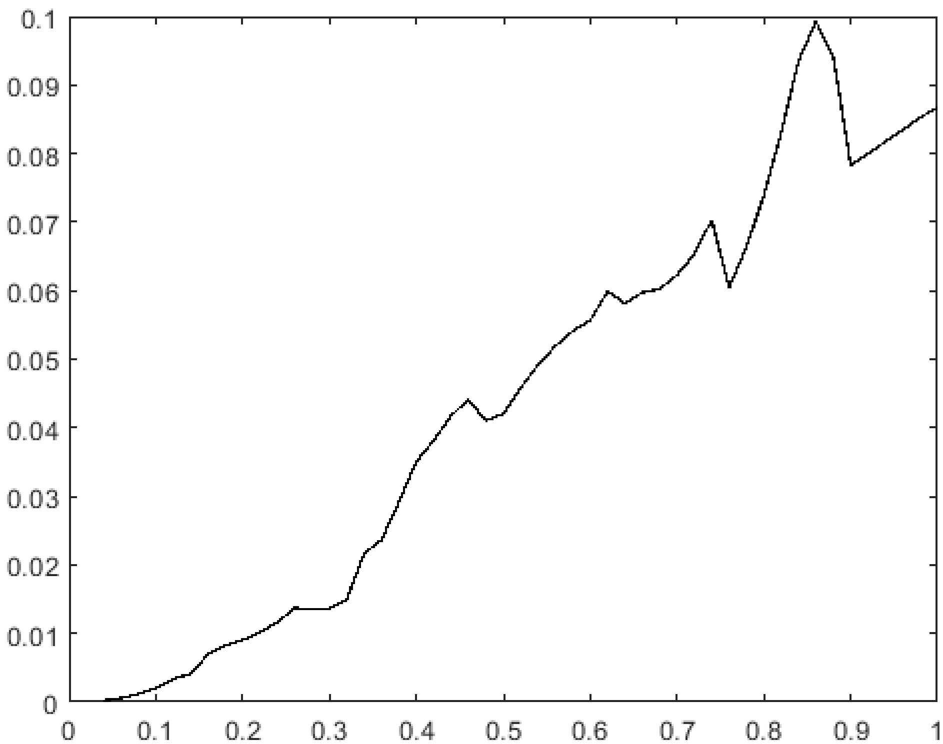
```
figure(15)
plot(S9.Period,S9.CDPSa,'k','LineWidth',1)
```



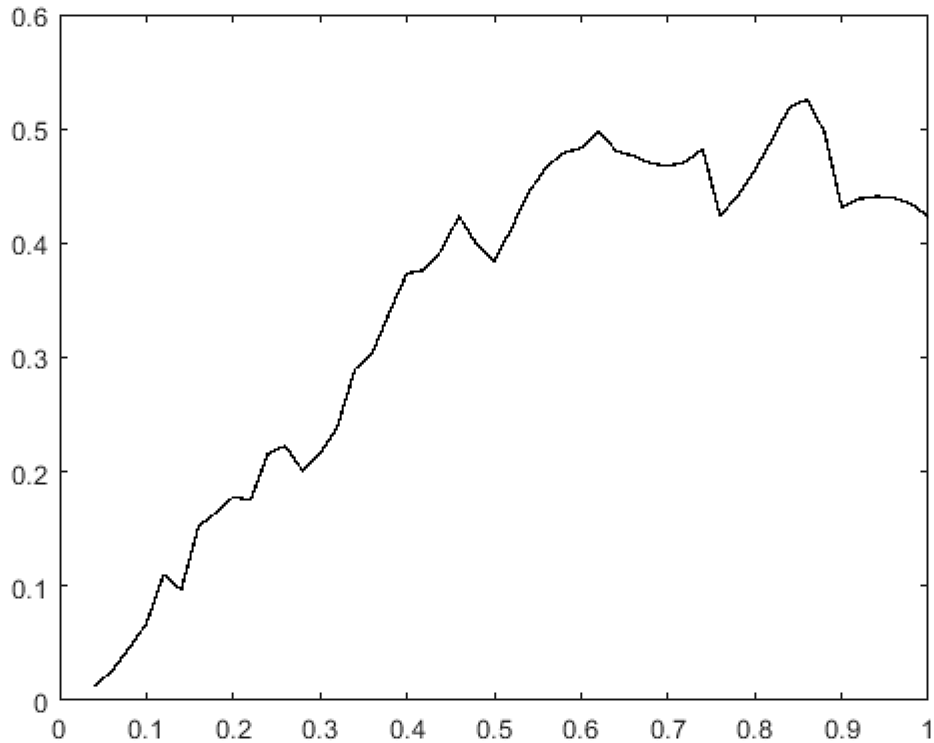

```
figure(16)
plot(S9.Period,S9.CDPSv,'k','LineWidth',1)
```



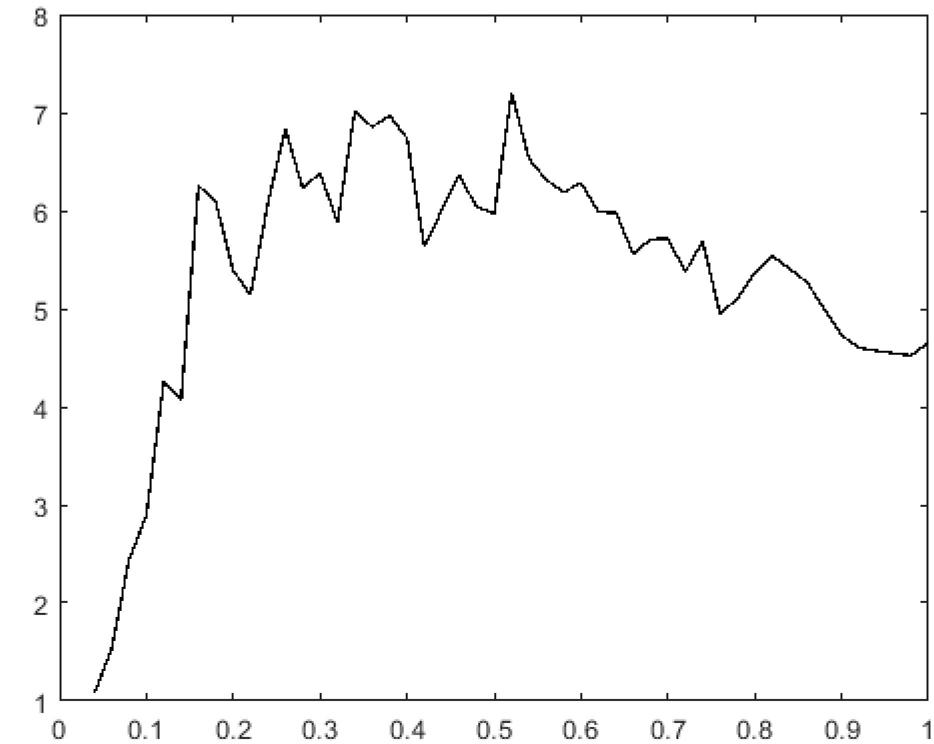

```
figure(17)
plot(S9.Period,S9.CDSd,'k','LineWidth',1)
```



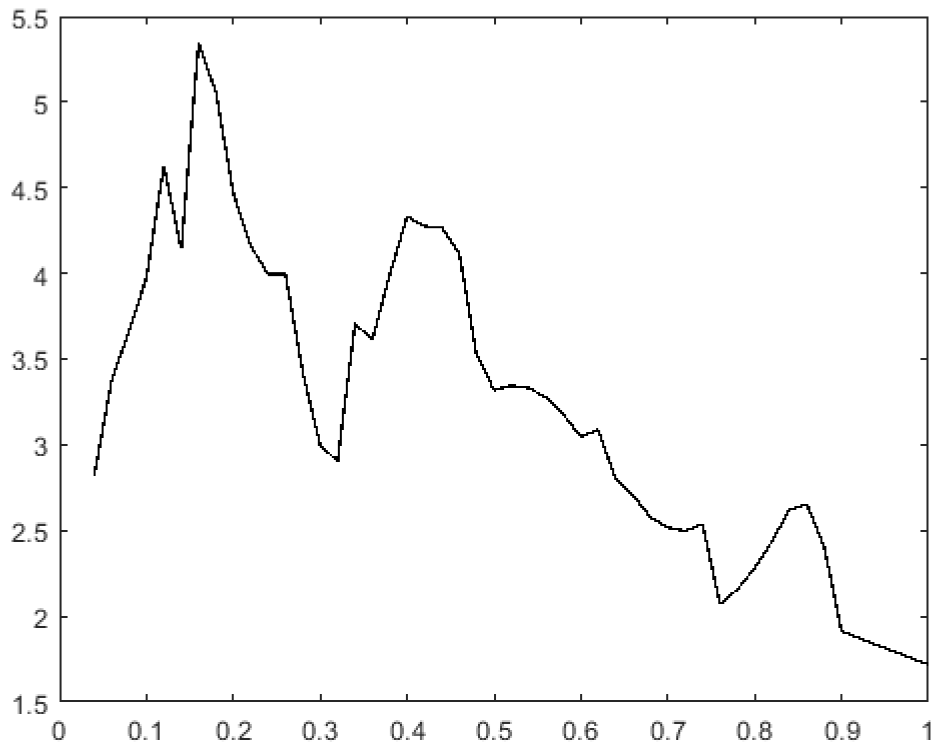
```
figure(18)
plot(S9.Period,S9.CDSv,'k','LineWidth',1)
```



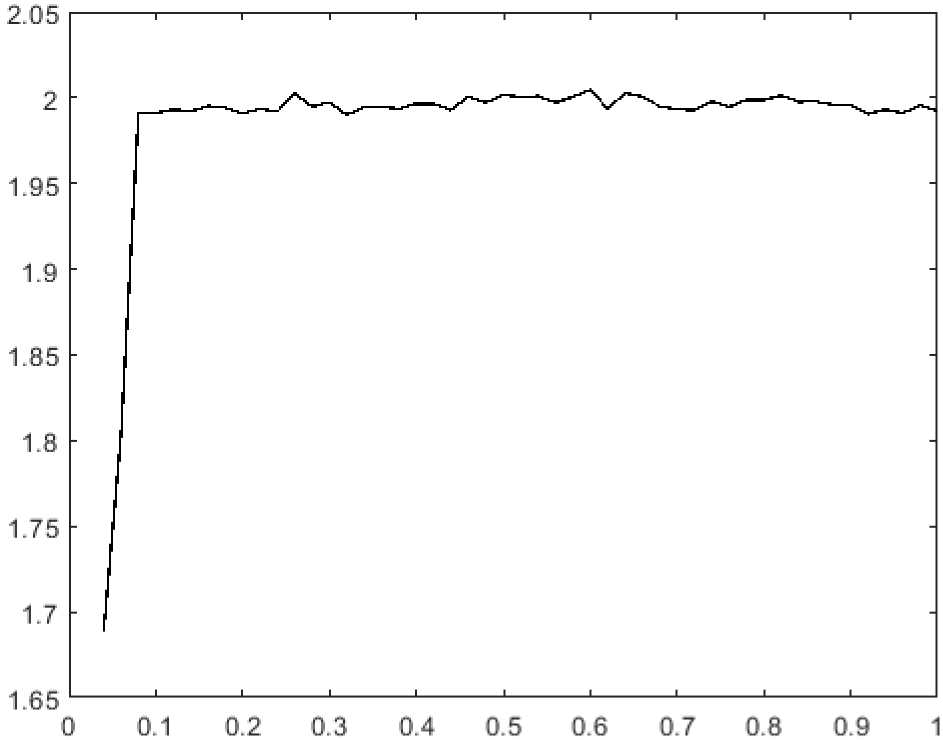

```
figure(19)
plot(S9.Period,S9.CDSa,'k','LineWidth',1)
```



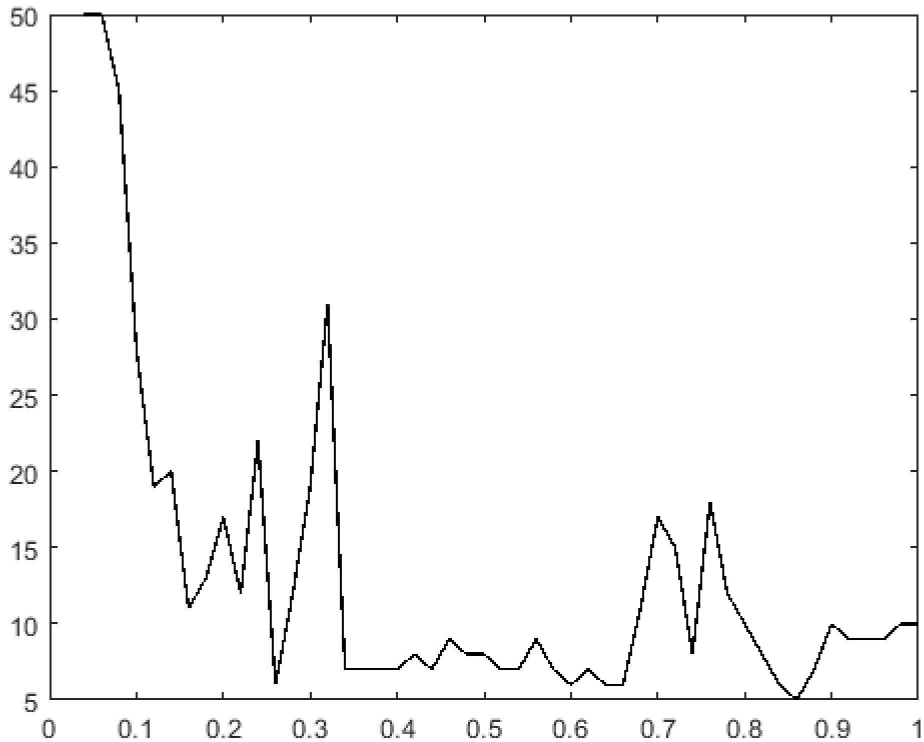

```
figure(20)
plot(S9.Period,S9.fyK,'k','LineWidth',1)
```



```
figure(21)
plot(S9.Period,S9.muK,'k','LineWidth',1)
```



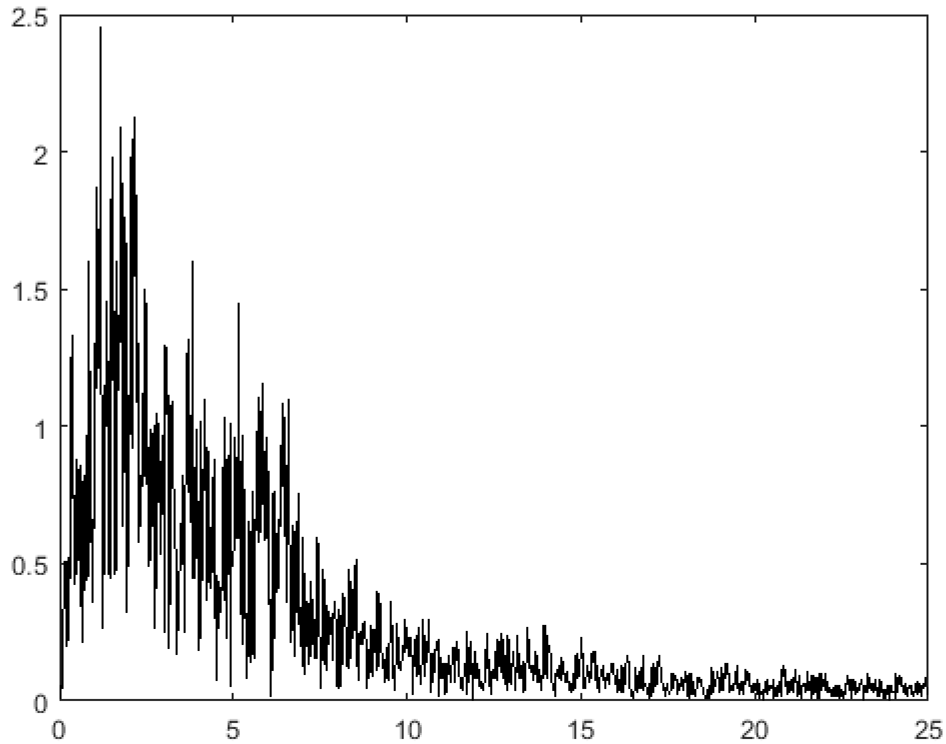

```
figure(22)
plot(S9.Period,S9.iterK,'k','LineWidth',1)
```




### Fourier amplitude spectrum and mean period

```
sw='fas';
S10=OpenSeismoMatlab(dt,xgtt,sw);
```

```
figure(23)
plot(S10.freq,S10.FAS,'k','LineWidth',1)
```




```
S10.Tm
```

```
ans =

   0.533389590787339
```

```
S10.Fm
```

```
ans =

   3.293755163661824
```

## Copyright




- Major, Infrastructure Engineer, Hellenic Air Force
- Civil Engineer, M.Sc., Ph.D.
- Email: gpapazafeiropoulos@yahoo.gr


*Published with MATLAB® R2017b*