OpenSeismoMatlab

Software for strong ground motion data processing

-

Contents

Help

Help section of the OpenSeismoMatlab functions

-

baselineCorr.m

Baseline correction of acceleration time history

-

BLKIN.m

Bilinear kinematic hysteretic model with elastic viscous damping

-

CDReSp.m

Constant Ductility Response Spectra

-

CSReSp.m

Constant Strength Response Spectra

-

DRHA.m

Dynamic Response History Analysis

-

FASp.m

Fourier amplitude spectrum

-

HalfStep.m

Reproduce signal with half time step

-

IDA.m

Incremental Dynamic Analysis

-

LEReSp.m

Linear Elastic Response Spectra

-

LIDA.m

Linear Implicit Dynamic Analysis

-

NLIDABLKIN.m

Non Linear Implicit Dynamic Analysis of a BiLinear KINematic hardening hysteretic structure with elastic damping

-

OpenSeismoMatlab.m

Seismic parameters and processing of an acceleration time history

-

PulseDecomp.m

Pulse decomposition of a velocity time history

-

RPSReSp.m

Rigid Plastic Sliding Response Spectra

-

Examples

Examples of various OpenSeismoMatlab applications

-

example_baselineCorr

Perform baseline correction

-

example_CDReSp

Calculate constant ductility response spectra of an artificially generated acceleration time history

-

example_Constant_ductility_response_spectra

Calculate constant ductility response spectra of a suite of ground motions

-

example_Fourier_spectra

Calculate Fourier spectra of a suite of ground motions

-

example_general

Test all functionalities of OpenSeismoMatlab in a single script

-

example_LEReSp

Calculate linear elastic response spectra of an artificially generated acceleration time history

-

example_Linear_elastic_response_spectra.m

Calculate the linear elastic response spectra of an earthquake suite

-

PulseDecomp.m

Pulse decomposition of a velocity time history

-

RPSReSp.m

Calculate rigid plastic sliding response spectra

-

example_Spectra_comparison_1.m

Comparison of elastic and constant ductility response spectra for mu=1

-

example_Spectra_comparison_2.m

Comparison of constant ductility response spectra for mu=2

-

Verification

Validation of OpenSeismoMatlab results against the literature

verification_CDRS

Verification of a constant ductility response spectrum

Reference: Chopra (2020)

-

verification_CSRS1

Verification of a constant strength response spectrum

Reference: Tena-Colunga, A. (1999)

-

verification_CSRS2

Verification of a constant strength response spectrum

Reference: Tena-Colunga, A. (1999)

-

verification_DRHA

Calculate linear dynamic response of a MDOF shear building

Reference: Chopra (2020)

-

verification_filter1

Verify the high pass Butterworth filter of OpenSeismoMatlab

Reference: Boore (2005)

-

verification_filter2

Verify the low- and high- pass Butterworth filter of OpenSeismoMatlab

Reference: Graizer (2012)

-

verification_Fourier_spectrum1

Verify the Fourier amplitude spectrum of the San Fernando earthquake (1971) - Component N11E

Reference: California Institute of Technology, EERL (1974)

-

verification_Fourier_spectrum2

Verify the Fourier amplitude spectrum of the San Fernando earthquake (1971) - Component N79W

Reference: California Institute of Technology, EERL (1974)

-

verification_IDA1

Verify the incremental dynamic analysis for an acceleration time history with given duration D_5_75 and spectral acceleration

Reference: Mashayekhi et al. (2020)

-

verification_IDA2

Verify the incremental dynamic analysis for an earthquake suite

Reference: Deng et al. (2017)

-

verification_IDA3

Verify the incremental dynamic analysis for the Loma Prieta earthquake (1989)

Reference: Vamvatsikos & Cornell (2002)

-

verification_IDA4

Verify the incremental dynamic analysis bias due to fitting of the capacity curve of a SDOF system with an elastoplastic bilinear fit according to FEMA-440

Reference: De Luca et al. (2011)

-

verification_LIDA

Calculate the dynamic response of a linear SDOF oscillator

Reference: Chopra (2020)

-

verification_LIDA_NLIDABLKIN

Compare the output of LIDA.m and NLIDABLKIN.m for a linear SDOF oscillator in terms of seismic input energy per unit mass.

Reference: None

-

verification_NLIDABLKIN

Verify the energy time history of SDOF oscillator

Reference: Chopra (2020)

-

verification_NLIDABLKIN2

Verify the energy time histories of a nonlinear SDOF oscillator

Reference: Uang, C. M., & Bertero, V. V. (1990)

-

verification_NLIDABLKIN3

Verify the strength ductility relation of an elastoplastic system

Reference: Mahin, S. A., & Lin, J. (1983)

-

verification_PulseDecomp1

Verify the extraction of the velocity pulse of a strong motion velocity time history

Reference: Baker, J. W. (2007)

-

verification_PulseDecomp2

Verify the extraction of the velocity pulses of a strong motion velocity time history and sum them to reconstruct the original ground motion

Reference: Shahi, S. K., & Baker, J. W. (2014)

-

verification_SIH1952

Calculate the spectral intensity as defined by Housner (1952)

Reference: Housner, G. W. (1952)

-

verification_SRS1

Calculate the rigid plastic sliding response spectrum

Reference: Garini, E., & Gazetas, G. (2016)

-

verification_SRS2

Calculate the rigid plastic sliding response spectrum

Reference: Paglietti, A., & Porcu, M. C. (2001)

-

Copyright

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

doc 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. First order fitting (straight line) is
     performed on the raw acceleration time history and the fitting line
     is subtracted from the raw acceleration time history, giving thus the
     corrected acceleration time history.

 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=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)

__________________________________________________________________________
 Copyright (c) 2018-2023
     George Papazafeiropoulos
```

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

doc BLKIN

Documentation of the BLKIN function.

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

```
 Bilinear kinematic 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)
```

doc CDReSp

Documentation of the CDReSp function.

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

```
  Constant Ductility Response Spectra

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

  Description
      The constant ductility response spectra for a given time-history of
      constant time step, eigenperiod range, viscous damping ratio and
      ductility are computed. See section 7.5 in Chopra (2012) and the
      notes "Inelastic Response Spectra" (CEE 541. Structural Dynamics) by
      Henri P. Gavin.

  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.
      TOL [double(1 x 1)] is the tolerance for convergence for the target
          ductility.
      PYSF [double(1 x 1)] is the post-yield stiffness factor, i.e. the
          ratio of the postyield stiffness to the initial stiffness. PYSF=0
          is not recommended for simulation of an elastoplastic system. A
          small positive value is always suggested. PYSF is ignored if
          MU=1.
      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.
      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).
    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
    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
    figure()
    plot(t,xgtt)
    T=(0.04:0.04:4)';
    ksi=0.05;
    mu=2;
    n=50;
    tol=0.01;
    pysf=0.1;
    dtTol=0.02;
    AlgID='U0-V0-Opt';
    rinf=1;
    maxtol=0.01;
    jmax=200;
    dak=eps;
    [CDPSa,CDPSv,CDSd,CDSv,CDSa,fyK,muK,iterK]=CDReSp(dt,xgtt,T,ksi,...
        mu,n,tol,pysf,dtTol,AlgID,rinf,maxtol,jmax,dak);
    figure()
    plot(T,CDSd)
    figure()
    plot(T,fyK)
    figure()
    plot(T,muK)
    figure()
    plot(T,iterK)
```

doc CSReSp

Documentation of the CSReSp function.

```
helpFun('CSReSp')
```

```
 Constant Strength Response Spectra

 [SMU,SD,SV,SA,SEY,SED]=CSRESP(DT,XGTT,T,KSI,FYR,PYSF,DTTOL,...
     ALGID,RINF,MAXTOL,JMAX,DAK)

 Description
     The constant strength response spectra for a given time-history of
     constant time step, eigenperiod range, viscous damping ratio and
     yield strength ratio (yield shear to weight, V/W) 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.
     FYR [double(1 x 1)] is the yield strength ratio (yield shear to
         weight, V/W) for which the response spectra are calculated.
     PYSF [double(1 x 1)] is the post-yield stiffness factor, i.e. the
         ratio of the postyield stiffness to the initial stiffness. PYSF=0
         is not recommended for simulation of an elastoplastic system. A
         small positive value is always suggested.
     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.
     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).
    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
    SMU [double(1:numSDOFs x 1)] is the Spectral ductility.
    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.
    SEY [double(1:numSDOFs x 1)] is the Spectral yield energy.
    SED [double(1:numSDOFs x 1)] is the Spectral damping energy.

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
    figure()
    plot(t,xgtt)
    T=(0.04:0.04:4)';
    ksi=0.05;
    fyR=0.1;
    pysf=0.1;
    dtTol=0.02;
    AlgID='U0-V0-Opt';
    rinf=1;
    maxtol=0.01;
    jmax=200;
    dak=eps;
    [Smu,Sd,Sv,Sa,Sey,Sed]=CSReSp(dt,xgtt,T,ksi,fyR,pysf,dtTol,...
      AlgID,rinf,maxtol,jmax,dak);
    figure()
    plot(T,Smu)
    figure()
    plot(T,Sd)
    figure()
    plot(T,Sv)
    figure()
    plot(T,Sa)
    figure()
    plot(T,Sey)
    figure()
    plot(T,Sed)
```

doc DRHA

Documentation of the DRHA function.

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

```
  Dynamic Response History Analysis

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

  Description
      Calculate the dynamic response of a linear MDOF system using modal
      analysis. This function is part of the OpenSeismoMatlab software. It
      can be used as standalone, however attention is needed for the
      correctness of the input arguments, since no checks are performed in
      this function. See the example example_DRHA.m for more details about
      how this function can be implemented.

  Input parameters
      K [double(:inf x 1)] is the stiffness of the system.
      M [double(:inf x 1)] is the lumped masses of the structure.
      DT [double(1 x 1)] is the time step of the dynamic response history
          analysis
      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.
      U0 [double(:inf x 1)] is the initial displacement of the SDOF system.
      UT0 [double(:inf x 1)] is the initial velocity of the SDOF system.
      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
    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.
```

doc FASp

Documentation of the FASp function.

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

```
 Fourier amplitude spectrum

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

 Description
     Fourier amplitude spectrum of an acceleration time history.

 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) 2018-2023
     George Papazafeiropoulos
     Major, Infrastructure Engineer, Hellenic Air Force
     Civil Engineer, M.Sc., Ph.D.
     Email: gpapazafeiropoulos@yahoo.gr
 _________________________________________________________________________
```

doc 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)
 
__________________________________________________________________________
 Copyright (c) 2018-2023
     George Papazafeiropoulos
     Major, Infrastructure Engineer, Hellenic Air Force
     Civil Engineer, M.Sc., Ph.D.
     Email: gpapazafeiropoulos@yahoo.gr
 _________________________________________________________________________
```

Published with MATLAB® R2022b

doc IDA

Documentation of the IDA function.

```
helpFun('IDA')
```

```
 Incremental Dynamic Analysis

 [DM,IM]=IDA(DT,XGTT,T,LAMBDAF,IM_DM,M,UY,PYSF,KSI,ALGID,U0,UT0,...
     RINF,MAXTOL,JMAX,DAK)

 Description
     This function performs incremental dynamic analysis of a given
     acceleration time history and SDOF oscillator.

 Input parameters
     DT [double(1 x 1)] is the time step of the input acceleration time
         history XGTT.
     XGTT [double(:inf x 1)] is the input acceleration time history.
         numsteps is the length of the input acceleration time history.
     T [double(1 x 1)] contains the eigenperiod of the SDOF system for
         which the incremental dynamic analysis response curve is
         requested.
     LAMBDAF [double(:inf x 1)] contains the values of the scaling factor
         (lambda factor) for the incremental dynamic analysis.
     IM_DM [char(1 x :inf)] is the Intensity Measure (IM) - Damage Measure
         (DM) pair that is to be calculated from the incremental dynamic
         analysis. IM_DM can take one of the following values (strings are
         case insensitive):
         'SA_MU': Spectral acceleration-ductility
         'PGD_MU': Peak displacement-ductility
         'PGV_MU': Peak velocity-ductility
         'PGA_MU': Peak acceleration-ductility
         'SA_DISP': Spectral acceleration-displacement
         'PGD_DISP': Peak displacement-displacement
         'PGV_DISP': Peak velocity-displacement
         'PGA_DISP': Peak acceleration-displacement
         'SA_VEL': Spectral acceleration-velocity
         'PGD_VEL': Peak displacement-velocity
         'PGV_VEL': Peak velocity-velocity
         'PGA_VEL': Peak acceleration-velocity
         'SA_ACC': Spectral acceleration-acceleration
         'PGD_ACC': Peak displacement-acceleration
         'PGV_ACC': Peak velocity-acceleration
         'PGA_ACC': Peak acceleration-acceleration
     M [double(1 x 1)] is the mass of the SDOF oscillator.
     UY [double(1 x 1)] is the yield displacement of the SDOF oscillator.
     PYSF [double(1 x 1)] is the post-yield stiffness factor, i.e. the
         ratio of the postyield stiffness to the initial stiffness. PYSF=0
         is not recommended for simulation of an elastoplastic system. A
         small positive value is always suggested. PYSF is ignored if
         MU=1.
     KSI [double(1 x 1)] is the fraction of critical viscous damping.
     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 of the SDOF
         oscillator.
     UT0 [double(1 x 1)] is the initial velocity of the SDOF oscillator.
     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.
     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
     DM [double(:inf x 1)] is the Damage Measure.
     IM [double(:inf x 1)] is the Intensity Measure.

 Example
     eqmotions={'elcentro'};
     data=load([eqmotions{1},'.dat']);
     t=data(:,1);
     dt=t(2)-t(1);
     xgtt=data(:,2);
     sw='ida';
     T=1;
     lambdaF=logspace(log10(0.001),log10(10),100);
     IM_DM='Sa_disp';
     m=1;
     uy = 0.082*9.81/(2*pi/T)^2;
     pysf=0.01;
     ksi=0.05;
     S5=OpenSeismoMatlab(dt,xgtt,sw,T,lambdaF,IM_DM,m,uy,pysf,ksi);
     figure()
     plot(S5.DM*1000,S5.IM/9.81,'k','LineWidth',1)
     grid on
     xlabel('Displacement (mm)')
     ylabel('Sa(T1,5%)[g]')
     xlim([0,200])
     ylim([0,0.7])
```

doc LEReSp

Documentation of the LEReSp function.

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

```
 Linear Elastic Response Spectra

 [PSA,PSV,SD,SV,SA,SIEV]=LERESP(DT,XGTT,T,KSI,DTTOL,...
     ALGID,RINF)

 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. These spectra include the spectral
     acceleration, spectral velocity, spectral displacement,
     pseudoacceleration, pseudovelocity, absolute equivalent input energy
     velocity and relative equivalent input energy velocity. This function
     is part of the OpenSeismoMatlab software. It can be used as
     standalone, however attention is needed for the correctness of the
     input arguments, since no checks are performed in this function. See
     the example example_LEReSp.m for more details about how this function
     can be implemented.

 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.
     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.
    SIEV [double(1:numSDOFs x 1)] is the equivalent relative input
        energy velocity. See: {Uang, C. M., & Bertero, V. V. (1990).
        Evaluation of seismic energy in structures. Earthquake
        engineering & structural dynamics, 19(1), 77-90} for more
        details.

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;
    dtTol=0.02;
    AlgID='U0-V0-Opt';
    rinf=1;
    [PSa,PSv,Sd,Sv,Sa,Siev]=LEReSp(dt,xgtt,T,ksi,dtTol,...
        AlgID,rinf);
```

doc LIDA

Documentation of the LIDA function.

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

```
 Linear Implicit Dynamic Analysis

 [U,UT,UTT,EI] = LIDA(DT,XGTT,OMEGA,KSI,U0,UT0,ALGID,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. This function is part of
     the OpenSeismoMatlab software. It can be used as standalone, however
     attention is needed for the correctness of the input arguments, since
     no checks are performed in this function. See the example
     example_LIDA.m for more details about how this function can be
     implemented.

 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.
     U0 [double(1 x 1)] is the initial displacement of the SDOF system.
     UT0 [double(1 x 1)] is the initial velocity of the SDOF system.
     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
    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
    EI [double(1:nstep x 1)] is the time-history of the seismic input
        energy per unit mass. See: {Uang, C. M., & Bertero, V. V. (1990).
        Evaluation of seismic energy in structures. Earthquake
        engineering & structural dynamics, 19(1), 77-90} for more
        details.

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=text{1,2};
    Tn=1;
    omega=2*pi/Tn;
    ksi=0.02;
    u0=0;
    ut0=0;
    AlgID='U0-V0-Opt';
    rinf=1;
    [u,ut,utt] = LIDA(dt,xgtt,omega,ksi,u0,ut0,AlgID,rinf);
    D=max(abs(u))/0.0254

_________________________________________________________________________
Copyright (c) 2018-2023
    George Papazafeiropoulos
    Major, Infrastructure Engineer, Hellenic Air Force
    Civil Engineer, M.Sc., Ph.D.
    Email: gpapazafeiropoulos@yahoo.gr
_________________________________________________________________________
```

doc 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

 ______________________________________________________________________
 Copyright (c) 2018-2023
    George Papazafeiropoulos
    Major, Infrastructure Engineer, Hellenic Air Force
    Civil Engineer, M.Sc., Ph.D.
    Email: gpapazafeiropoulos@yahoo.gr
 ______________________________________________________________________
```

Published with MATLAB® R2022b

doc OpenSeismoMatlab

Documentation of the OpenSeismoMatlab function.

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

```
  Seismic parameters and processing of an acceleration time history

  Syntax
      PARAM=OpenSeismoMatlab(DT,XGTT,SW,__)
      PARAM=OpenSeismoMatlab(DT,XGTT,'PGA')
      PARAM=OpenSeismoMatlab(DT,XGTT,'EPGA')
      PARAM=OpenSeismoMatlab(DT,XGTT,'PGV')
      PARAM=OpenSeismoMatlab(DT,XGTT,'PGD')
      PARAM=OpenSeismoMatlab(DT,XGTT,'ARIAS')
      PARAM=OpenSeismoMatlab(DT,XGTT,'SIH1952')
      PARAM=OpenSeismoMatlab(DT,XGTT,'SINH1984')
      PARAM=OpenSeismoMatlab(DT,XGTT,'TIMEHIST',BASELINESW)
      PARAM=OpenSeismoMatlab(DT,XGTT,'SINCRESAMPLE',DTI)
      PARAM=OpenSeismoMatlab(DT,XGT,'PULSEDECOMP',WNAME,TPMIN,TPMAX,...
          NSCALES)
      PARAM=OpenSeismoMatlab(DT,XGTT,'ELRS',T,KSI,ALGID,RINF,DTTOL)
      PARAM=OpenSeismoMatlab(DT,XGTT,'RPSRS',CF,ALGID,RINF,MAXTOL,JMAX,DAK)
      PARAM=OpenSeismoMatlab(DT,XGTT,'CDRS',T,KSI,MU,PYSF,DTTOL,ALGID,...
          RINF,MAXTOL,JMAX,DAK)
      PARAM=OpenSeismoMatlab(DT,XGTT,'CSRS',T,KSI,FYR,PYSF,DTTOL,ALGID,...
          RINF,MAXTOL,JMAX,DAK)
      PARAM=OpenSeismoMatlab(DT,XGTT,'FAS')
      PARAM=OpenSeismoMatlab(DT,XGTT,'BUTTERWORTHHIGH',BORDER,FLC)
      PARAM=OpenSeismoMatlab(DT,XGTT,'BUTTERWORTHLOW',BORDER,FHC)
      PARAM=OpenSeismoMatlab(DT,XGTT,'IDA',T,LAMBDAF,IM_DM,M,UY,PYSF,...
          KSI,ALGID,U0,UT0,RINF,MAXTOL,JMAX,DAK)
      Omit or set as empty ([]) the input arguments for which default
      values are desired.

  Description
      This function calculates the seismic parameters, develops various
      spectra and performs various analyses from an acceleration time
      history. More specifically, it calculates the following:
      1) Peak ground acceleration
      2) Effective peak ground acceleration
      3) Peak ground velocity
      4) Peak ground displacement
      5) Total cumulative energy and normalized cumulative energy vs time
      6) Significant duration D_5_95 according to Trifunac & Brady (1975)
      7) Significant duration D_5_75
      8) Total Arias intensity (Ia)
      9) Velocity time history (with baseline correction or not)
      10) Displacement time history (with baseline correction or not)
      11) Resampled acceleration time history (i.e. the input acceleration
         time history with modified time step size)
      12) Linear elastic pseudo-acceleration response spectrum
      13) Linear elastic pseudo-velocity response spectrum
      14) Linear elastic displacement response spectrum
      15) Linear elastic velocity response spectrum
      16) Linear elastic acceleration response spectrum
```

```
    17) Rigid plastic sliding displacement response spectrum
    18) Rigid plastic sliding velocity response spectrum
    19) Rigid plastic sliding acceleration response spectrum
    20) Constant ductility displacement response spectrum
    21) Constant ductility velocity response spectrum
    22) Constant ductility acceleration response spectrum
    23) Fourier amplitude spectrum
    24) Mean period (Tm)
    25) Lowpass Butterworth-filtered acceleration time history
    26) Highpass Butterworth-filtered acceleration time history
    27) Incremental Dynamic Analysis (IDA) of SDOF system excited with
        the input acceleration time history
    28) Spectral intensity according to Housner (1952)
    29) Spectral intensity according to Nau & Hall (1984)
    Depending on the value of SW, which determines the type of analysis
    that OpenSeismoMatlab performs, various additional parameters are
    needed as input by the user. All possible syntaxes appear above.

Input parameters
    DT [double(1 x 1)] is the size of the time step of the input
        acceleration time history xgtt.
    XGTT [double(:inf x 1)] is the input acceleration time history.
    XGT [double(:inf x 1)] is the input velocity time history.
    SW [char(1 x :inf)] is a string which determines which parameters,
        spectras or analyses 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.
        'SINCRESAMPLE': the acceleration time history with modified time
            step size is calculated through sinc resampling.
        '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
            D_5_95 according to Trifunac & Brady (1975), significant
            duration D_5_75 and Arias intensity are calculated.
        'PULSEDECOMP': the (velocity) time history is decomposed into a
            pulse and a residual motion
        'ELRS': The linear elastic response spectra and pseudospectra are
            calculated.
        'RPSRS': The rigid plastic sliding response spectra are
            calculated.
        'CDRS': The constant ductility response spectra are calculated.
        'CSRS': The constant strength response spectra are calculated.
        'IDA': Incremental Dynamic Analysis of an elastoplastic SDOF
            system excited by the input acceleration time history is
            performed.
        'FAS': The Fourier amplitude spectrum and the mean period are
            calculated.
        'BUTTERWORTHHIGH': The high-pass Butterworth filtered
            acceleration time history is calculated.
        'BUTTERWORTHLOW': The low-pass Butterworth filtered
            acceleration time history is calculated.
        'EPGA': Effective Peak Ground Acceleration.
        'SIH1952': Spectral Intensity according to Housner (1952).
        'SINH1984': Spectral Intensity according to Nau & Hall (1984).

Additional required parameters
    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
        Default value 'U0-V0-Opt'.
    BASELINESW [logical(1 x 1)] determines if baseline correction will be
        applied for the calculation of the various time histories.
    BORDER [double(1 x 1)] is the order of the Butterworth filter that is
        applied for filtering of the acceleration time history.
    CF [double(1:numSDOFs x 1)] contains the values of the Coulomb
        friction coefficient for which the response spectra are
        requested. numSDOFs is the number of SDOF oscillators being
        analysed to produce the spectra.
    DAK [double(1 x 1)] is the infinitesimal acceleration for the
        calculation of the derivetive required for the convergence of the
        Newton-Raphson iteration.
    DTI [double(1 x 1)] is the new time step size for resampling of the
        input acceleration time history.
    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 0.02.
    FHC [double(1 x 1)] is the high cutoff frequency for the low-pass
        Butterworth filter.
    FLC [double(1 x 1)] is the low cutoff frequency for the high-pass
        Butterworth filter.
    FYR [double(1 x 1)] is the yield strength ratio (yield shear to
        weight, V/W).
    IM_DM [char(1 x :inf)] is the Intensity Measure (IM) - Damage Measure
```

```
        (DM) pair that is to be calculated from the incremental dynamic
        analysis. IM_DM can take one of the following values (strings are
        case insensitive):
        'SA_MU': Spectral acceleration-ductility
        'PGD_MU': Peak displacement-ductility
        'PGV_MU': Peak velocity-ductility
        'PGA_MU': Peak acceleration-ductility
        'SA_DISP': Spectral acceleration-displacement
        'PGD_DISP': Peak displacement-displacement
        'PGV_DISP': Peak velocity-displacement
        'PGA_DISP': Peak acceleration-displacement
        'SA_VEL': Spectral acceleration-velocity
        'PGD_VEL': Peak displacement-velocity
        'PGV_VEL': Peak velocity-velocity
        'PGA_VEL': Peak acceleration-velocity
        'SA_ACC': Spectral acceleration-acceleration
        'PGD_ACC': Peak displacement-acceleration
        'PGV_ACC': Peak velocity-acceleration
        'PGA_ACC': Peak acceleration-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.
    KSI [double(1 x 1)] is the fraction of critical viscous damping.
    LAMBDAF [double(:inf x 1)] contains the values of the scaling factor
        (lambda factor) for the incremental dynamic analysis.
    M [double(1 x 1)] is the mass of the SDOF oscillator.
    MAXTOL [double(1 x 1)] is the maximum tolerance of convergence of the
        Full Newton Raphson method for numerical computation of
        acceleration.
    MU [double(1 x 1)] is the specified ductility for which the constant
        ductility response spectra are calculated.
    NSCALES [double(1 x 1)] is the number of pulse period values between
        TPMIN and TPMAX to be considered for the continuous 1-D wavelet
        transform of XGT
    PYSF [double(1 x 1)] is the post-yield stiffness factor, i.e. the
        ratio of the postyield stiffness to the initial stiffness. PYSF=0
        is not recommended for simulation of an elastoplastic system; a
        small positive value is always suggested due to numerical
        reasons. PYSF is ignored if MU=1. Default value 0.01.
    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.
    T [double(:inf x 1)] contains the values of eigenperiods for
        which the response spectra are requested. Its length is the
        number of SDOF oscillators being analysed to produce the spectra.
        T must be a vector if SW='ELRS' or SW='CDRS'. T must be scalar if
        SW='IDA'.
    TPMAX [double(1 x 1)] is the maximum pulse period to be considered
        for the continuous 1-D wavelet transform of XGT
    TPMIN [double(1 x 1)] is the minimum pulse period to be considered
        for the continuous 1-D wavelet transform of XGT
    U0 [double(1 x 1)] is the initial displacement of the SDOF
        oscillator.
    UT0 [double(1 x 1)] is the initial velocity of the SDOF oscillator.
    UY [double(1 x 1)] is the yield displacement of the SDOF oscillator.
    WNAME [char(1 x :inf)] is the wavelet family short name to be used
        for the decomposition of the velocity time history. See the
        Matlab function waveinfo.m for more details.

Output parameters
    PARAM (structure) has the following fields:
```

```
PARAM.acc [double(:inf x 1)] Acceleration time history
PARAM.arias [double(1 x 1)] Total Arias intensity (Ia)
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.CDSa [double(:inf x 1)] Constant ductility acceleration
    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.disp [double(:inf x 1)] Displacement time history
param.DM [double(:inf x 1)] is the damage measure (DM) of the
    incremental dynamic analysis
PARAM.Ecum [double(1 x 1)] Total cumulative energy
PARAM.EcumTH [double(:inf x 1)] Normalized cumulative
    energy vs time
PARAM.EPGA [double(1 x 1)] Effective peak ground acceleration
PARAM.FAS [double(1:2^(nextpow2(length(xgtt))-1) x 1)] Fourier
    amplitude spectrum
PARAM.Fm [double(1 x 1)] Mean frequency (Fm)
PARAM.fyK [double(:inf x 1)] yield limit that each SDOF must have
    in order to attain ductility equal to PARAM.muK.
param.IM [double(:inf x 1)] is the intensity measure (IM) of the
    incremental dynamic analysis
PARAM.iterK [double(:inf x 1)] number of iterations needed for
    convergence for each period (each SDOF).
PARAM.muK [double(:inf x 1)] achieved ductility for each period
    (each SDOF).
PARAM.PGA [double(1 x 1)] Peak ground acceleration
PARAM.PGD [double(1 x 1)] Peak ground displacement
PARAM.PGV [double(1 x 1)] Peak ground velocity
PARAM.PredPeriod [double(1 x 1)] Predominant period of the PSa
    spectrum
PARAM.PredPSa [double(1 x 1)] Predominant acceleration of the PSa
    spectrum
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.pulseTH [double(:inf x 1)] Time history of the pulse
    contained in velocity time history
PARAM.resTH [double(:inf x 1)] Time history of the residual
    motion after subtracting the pulse from the velocity time
    history
PARAM.RPSSa [double(:inf x 1)] Rigid plastic sliding response
    spectral acceleration
PARAM.RPSSd [double(:inf x 1)] Rigid plastic sliding response
    spectral displacement
PARAM.RPSSv [double(:inf x 1)] Rigid plastic sliding response
    spectral velocity
PARAM.Sa [double(:inf x 1)] Linear elastic acceleration response
    spectrum
PARAM.Sd [double(:inf x 1)] Linear elastic displacement response
    spectrum
PARAM.SI [double(1 x 1)] Spectral intensity
PARAM.Siev [double(:inf x 1)] Linear elastic relative input
    energy equivalent velocity spectrum
PARAM.Sv [double(:inf x 1)] Linear elastic velocity response
```

```
        spectrum
    PARAM.t_5_75 [double(1 x 2)] Time instants at which 5% and 75% of
        cumulative energy have occurred
    PARAM.t_5_95 [double(1 x 2)] Time instants at which 5% and 95% of
        cumulative energy have occurred
    PARAM.Td_5_75 [double(1 x 1)] Time between when 5% and 75% of
        cumulative energy has occurred
    PARAM.Td_5_95 [double(1 x 1)] Time between when 5% and 95% of
        cumulative energy has occurred (significant duration
        according to Trifunac-Brady (1975))
    PARAM.time [double(:inf x 1)] Time
    PARAM.Tm [double(1 x 1)] Mean period (Tm)
    PARAM.Tp [double(1 x 1)] Period of the pulse
    PARAM.vel [double(:inf x 1)] Velocity time history
    PARAM.wavCoefs [double(1 x 1)] Coefficient for the extracted
        wavelet
    PARAM.wavScale [double(1 x 1)] Scale at which the largest wavelet
        was found
```

doc PulseDecomp

Documentation of the PulseDecomp function.

```
helpFun('PulseDecomp')
```

```
 Pulse decomposition of a velocity time history
 
 [PULSETH,RESTH,TP,WAVSCALE,WAVCOEFS]=PulseDecomp(DT,XGT,WNAME,TPMIN,...
     TPMAX,NSCALES)
 
 Description
     This function decomposes an input velocity time history record into a
     pulse motion and a residual motion. The shape of the pulse depends on
     the value of the WNAME parameter. To obtain the velocity time history
     of a given acceleration record use the command:
     PARAM=OpenSeismoMatlab(DT,XGTT,'TIMEHIST',BASELINESW)
     To extract more than one pulse wavelets from a velocity time history
     (let's say N), apply this function repetitively N times where at each
     time (except for the first) this function is applied to the residual
     motion of the previous time:
     [PULSETH1,RESTH1,~,~,~]=PulseDecomp(DT,XGT,WNAME,TPMIN,TPMAX,...
         NSCALES)
     [PULSETH2,RESTH2,~,~,~]=PulseDecomp(DT,RESTH1,WNAME,TPMIN,TPMAX,...
         NSCALES)
     etc.
 
 Input parameters
     DT [double(1 x 1)] is the time step of the input velocity time
         history XGT.
     XGT [double(1:numsteps x 1)] is the input velocity time history.
         numsteps is the length of the input velocity time history.
     WNAME [char(1 x :inf)] is the wavelet family short name to be used
         for the decomposition of the velocity time history. See the
         Matlab function waveinfo.m for more details.
     TPMIN [double(1 x 1)] is the minimum pulse period to be considered
         for the continuous 1-D wavelet transform of XGT
     TPMAX [double(1 x 1)] is the maximum pulse period to be considered
         for the continuous 1-D wavelet transform of XGT
     NSCALES [double(1 x 1)] is the number of pulse period values between
         TPMIN and TPMAX to be considered for the continuous 1-D wavelet
         transform of XGT
 
 Output parameters
     PULSETH [double(1:numsteps x 1)] is the velocity time history of the
         pulse contained in the input velocity time history. numsteps is
         the length of the input velocity time history.
     RESTH [double(1:numsteps x 1)] is the velocity time history of the
         residual motion after subtracting the time history of the pulse
         from the input velocity time history. numsteps is the length of
         the input velocity time history.
     TP [double(1 x 1)] is the period of the pulse which is extracted from
         the input velocity time history.
     WAVSCALE [double(1 x 1)] is the scale at which the largest wavelet
         was found.
     WAVCOEFS [double(1 x 1)] is the coefficient for the extracted
```

```
        wavelet.

Example
    rng(0)
    % Duration in seconds
    duration = 10;
    % Time step
    dt = 0.01;
    % Time vector
    t = linspace(0, duration, duration /dt)';
    % Number of sinusoidal components
    num_sin_components = 3;
    % Generate a random seismic signal using a combination of sinusoids
    xgt = zeros(length(t),1);
    for i = 1:num_sin_components
        % Generate the sinusoidal component
        s = rand * sin(2 * pi * 0.5*rand * t + rand);
        % Add the component to the seismic signal
        xgt = xgt + s;
    end
    % Setup wavelet parameters
    wname = 'db4';
    TpMin = 0.25;
    TpMax = 15;
    nScales = 50;
    % Apply pulse decomposition
    [pulseTH,resTH,Tp,wavScale,wavCoef] = ...
        PulseDecomp(dt,xgt,wname,TpMin,TpMax,nScales);
    % Plot the results
    np = length(xgt);
    time = dt:dt:dt*np;
    fig = figure();
    subplot(3,1,1)
    plot(time, xgt, '-k')
    legend('Original ground motion','location','northwest')
    ylabel('Velocity');
    set(gca, 'xticklabel', [])
    subplot(3,1,2)
    plot(time, pulseTH, '-r')
    legend('Extracted pulse','location','northwest')
    ylabel('Velocity');
    set(gca, 'xticklabel', [])
    subplot(3,1,3)
    plot(time, resTH , '-k')
    legend('Residual ground motion','location','northwest')
    hy = ylabel('Velocity');
    hx = xlabel('Time [s]');
```

doc RPSReSp

Documentation of the RPSReSp function.

```
helpFun('RPSReSp')
```

```
 Rigid Plastic Sliding Response Spectra

 [SD,SV,SA]=RPSReSp(DT,XGTT,CF,ALGID,RINF,MAXTOL,JMAX,DAK)

 Description
     The rigid plastic sliding response spectra for a given time-history
     of constant time step and a given Coulomb friction coefficient range
     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.
     CF [double(1:numSDOFs x 1)] contains the values of the Coulomb
         friction coefficient for which the response spectra are
         requested. numSDOFs is the number of SDOF oscillators being
         analysed to produce the spectra.
     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).
    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
    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.

Example
    dt=0.02;
    N=1;
    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
    figure()
    plot(t,xgtt)
    CF=linspace(0.05,0.5,1000)';
    AlgID='U0-V0-Opt';
    rinf=1;
    maxtol=0.01;
    jmax=200;
    dak=eps;
    [Sd,Sv,Sa]=RPSReSp(dt,xgtt,CF,AlgID,rinf,maxtol,jmax,dak);
    figure()
    plot(CF,Sd)
    figure()
    plot(CF,Sv)
    figure()
    plot(CF,Sa)
```

example baselineCorr

Apply baseline correction in OpenSeismoMatlab

Contents

Earthquake motion

Load earthquake data

```
fid=fopen('Imperial_Valley_El_Centro_9_EW.dat','r');
text=textscan(fid,'%f %f');
fclose(fid);
time=text{1,1};
xgtt1=text{1,2};
dt=time(2)-time(1);
```

Integrate without baseline correction

Calculate the velocity and displacement time histories

```
xgt1 = cumtrapz(time,xgtt1);
xg1 = cumtrapz(time,xgt1);
```

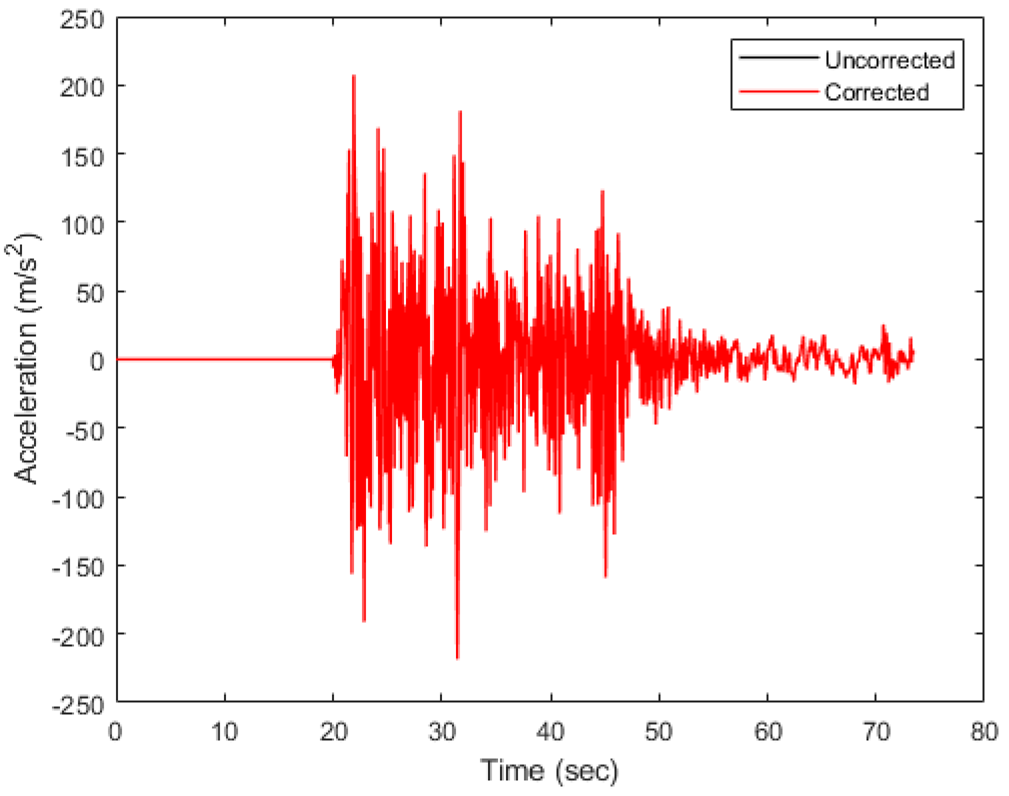
Integrate with baseline correction

Calculate the displacement, velocity and acceleration time histories

```
[xg2, xgt2, xgtt2] = baselineCorr(time,xgtt1);
```

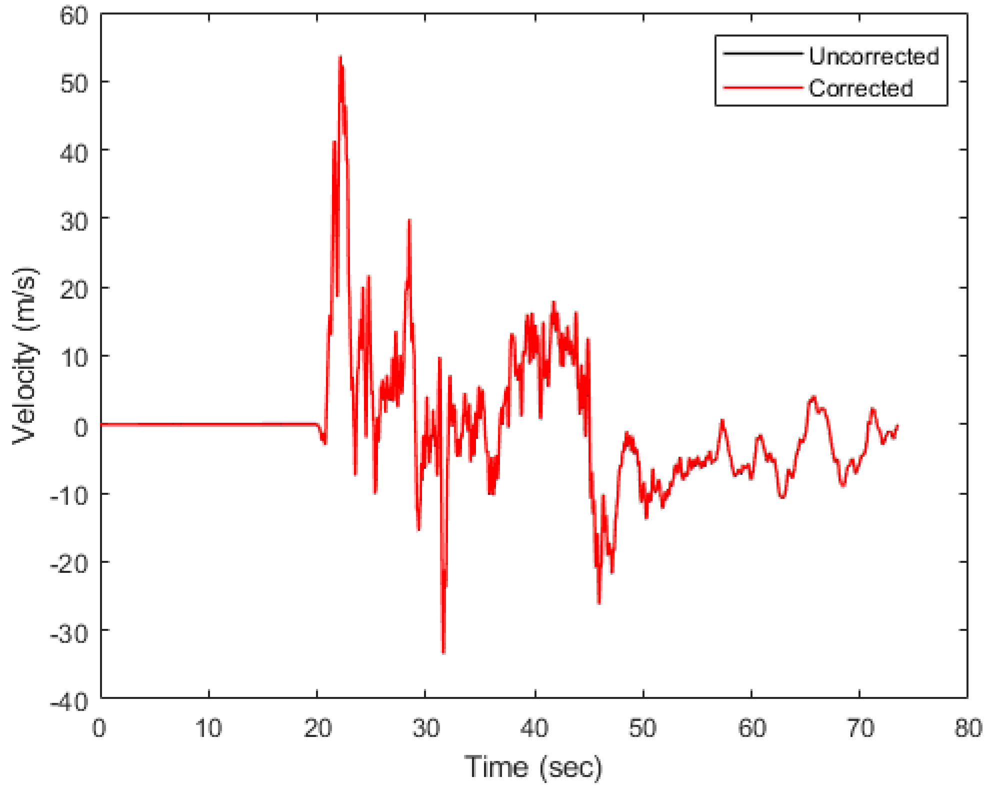
Plot corrected and uncorrected acceleration

```
figure()
plot(time,xgtt1,'k','LineWidth',1)
hold on
plot(time,xgtt2,'r','LineWidth',1)
hold off
ylabel('Acceleration (m/s^2)')
xlabel('Time (sec)')
legend('Uncorrected','Corrected')
drawnow;
pause(0.1)
```



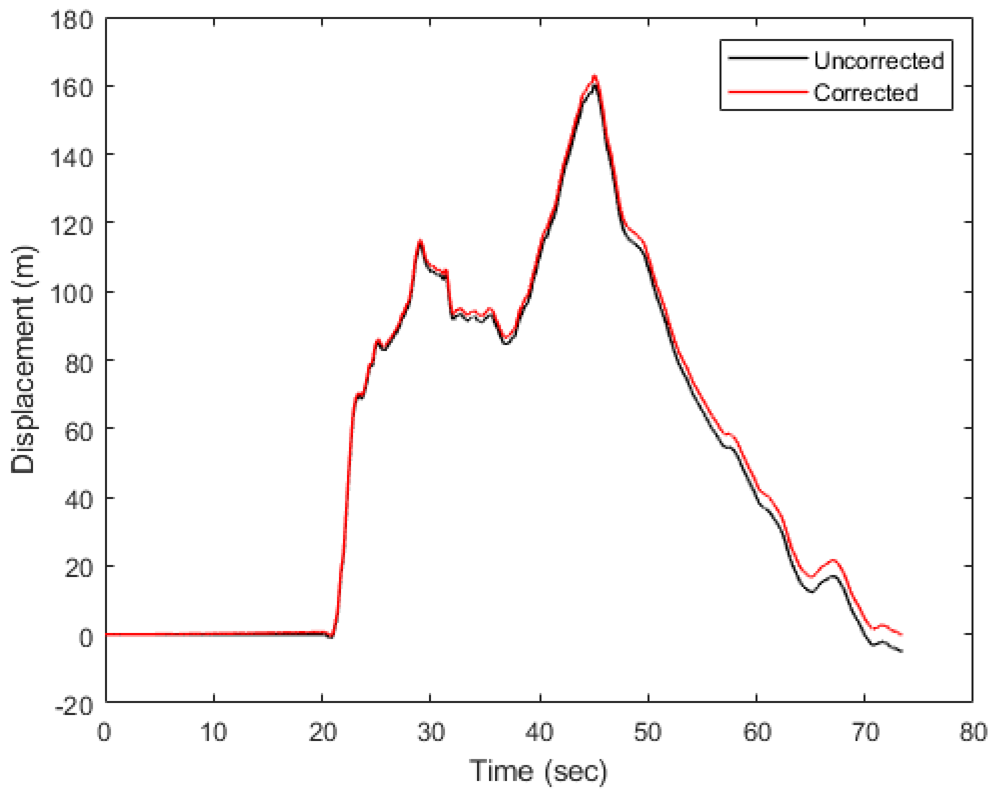
Plot corrected and uncorrected velocity

```
figure()
plot(time,xgt1,'k','LineWidth',1)
hold on
plot(time,xgt2,'r','LineWidth',1)
hold off
ylabel('Velocity (m/s)')
xlabel('Time (sec)')
legend('Uncorrected','Corrected')
drawnow;
pause(0.1)
```



Plot corrected and uncorrected displacement

```
figure()
plot(time,xg1,'k','LineWidth',1)
hold on
plot(time,xg2,'r','LineWidth',1)
hold off
ylabel('Displacement (m)')
xlabel('Time (sec)')
legend('Uncorrected','Corrected')
drawnow;
pause(0.1)
```



Copyright

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

example CDReSp

Calculate constant ductility response spectra in OpenSeismoMatlab

Contents

Earthquake motion

For reproducibility

```
rng(0)
```

Generate earthquake data

```
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
```

Setup parameters for CDReSp function

Eigenperiods

```
T=(0.04:0.04:4)';
```

Critical damping ratio

```
ksi=0.05;
```

Ductility

```
mu=2;
```

Maximum number of iterations

```
n=50;
```

Tolerance for convergence to target ductility

```
tol=0.01;
```

Post-yield stiffness factor

```
pysf=0.1;
```

Maximum ratio of the integration time step to the eigenperiod

```
dtTol=0.02;
```

Algorithm to be used for the time integration

```
AlgID='U0-V0-Opt';
```

Minimum absolute value of the eigenvalues of the amplification matrix

```
rinf=1;
```

Maximum tolerance of convergence for time integration algorithm

```
maxtol=0.01;
```

Maximum number of iterations per integration time step

```
jmax=200;
```

Infinitesimal acceleration

```
dak=eps;
```

Calculate spectra and pseudospectra

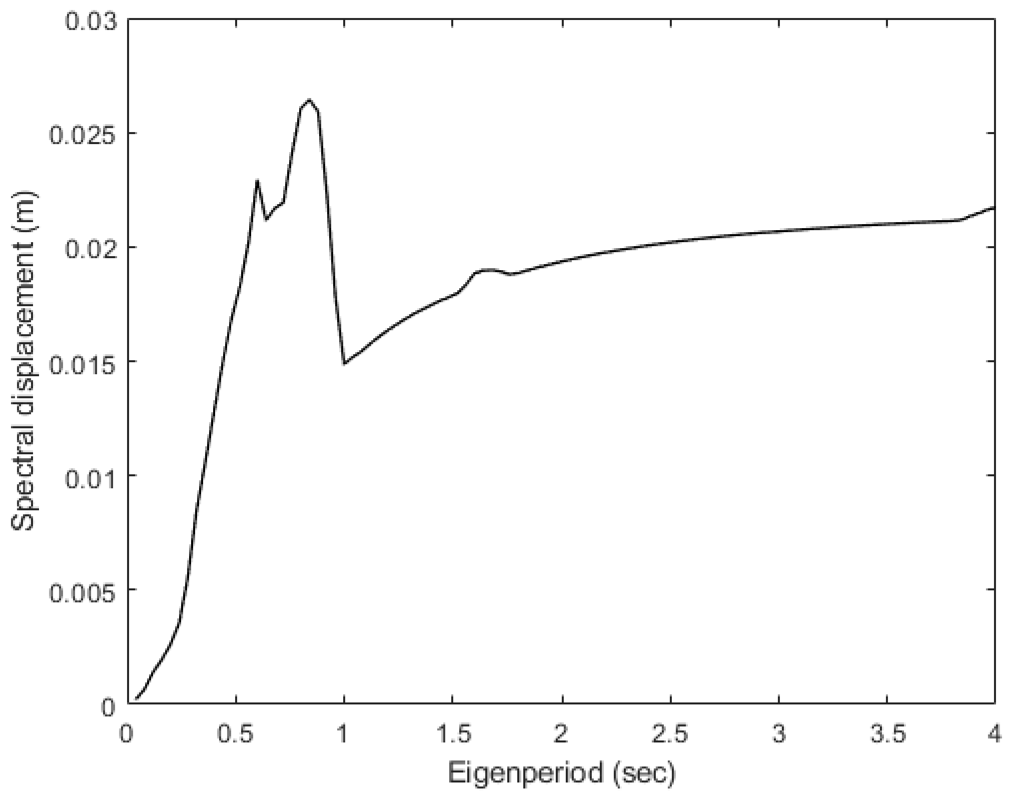
Apply CDReSp

```
[CDPSa,CDPSv,CDSd,CDSv,CDSa,fyK,muK,iterK]=CDReSp(dt,xgtt,T,ksi,...
    mu,n,tol,pysf,dtTol,AlgID,rinf,maxtol,jmax,dak);
```

Plot the spectra

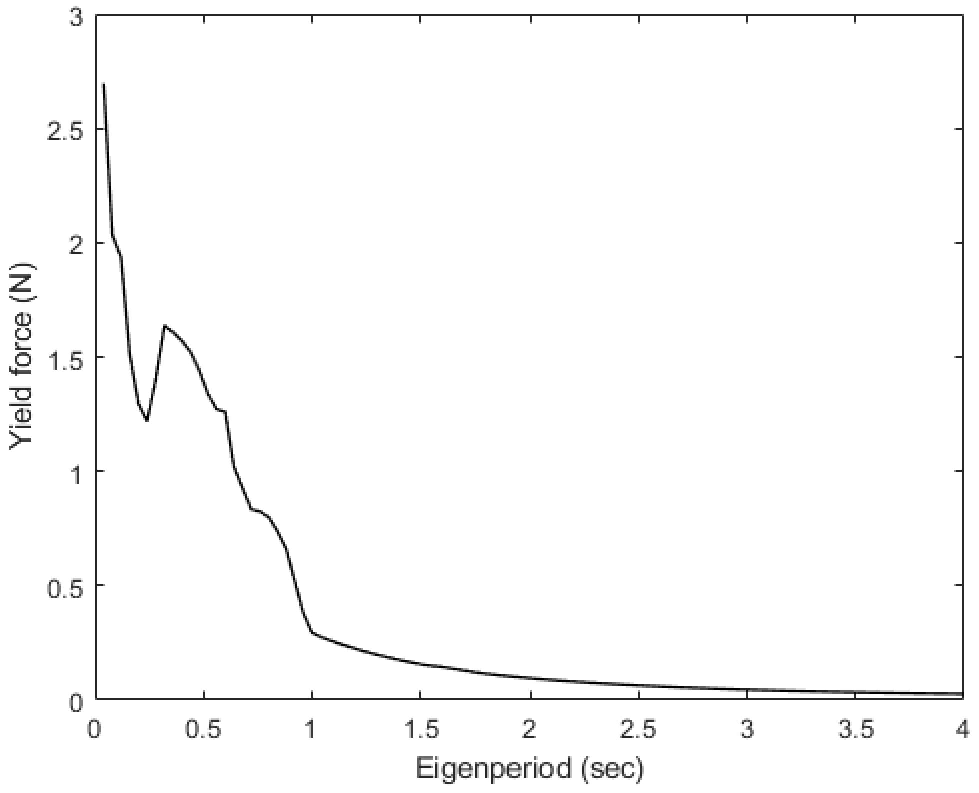
Constant ductility displacement spectrum

```
figure()
plot(T,CDSd,'k','LineWidth',1)
ylabel('Spectral displacement (m)')
xlabel('Eigenperiod (sec)')
drawnow;
pause(0.1)
```



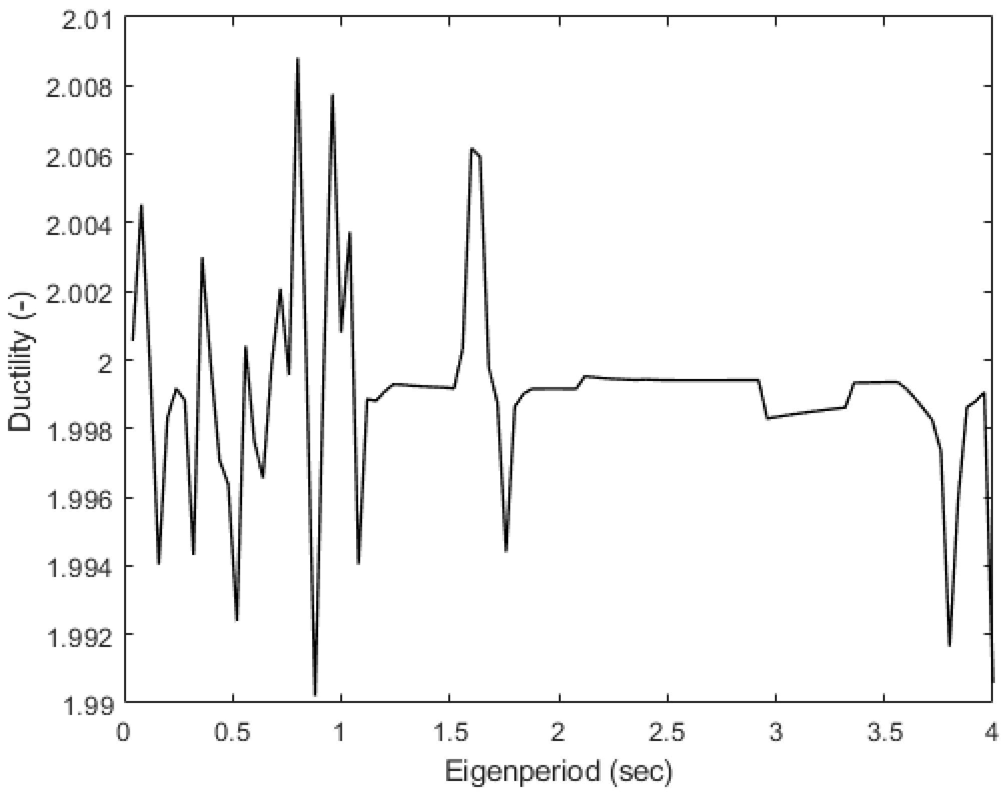
Constant ductility yield force spectrum

```
figure()
plot(T,fyK,'k','LineWidth',1)
ylabel('Yield force (N)')
xlabel('Eigenperiod (sec)')
drawnow;
pause(0.1)
```



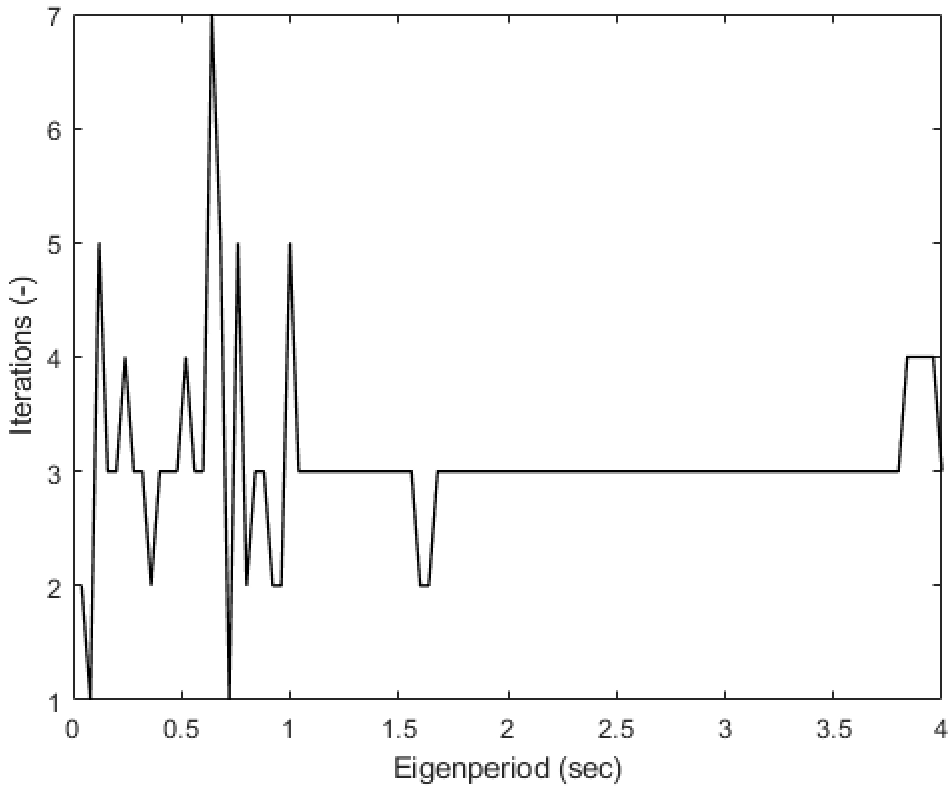
Achieved ductility

```
figure()
plot(T,muK,'k','LineWidth',1)
ylabel('Ductility (-)')
xlabel('Eigenperiod (sec)')
drawnow;
pause(0.1)
```



Iterations

```
figure()
plot(T,iterK,'k','LineWidth',1)
ylabel('Iterations (-)')
xlabel('Eigenperiod (sec)')
drawnow;
pause(0.1)
```



Copyright

Published with MATLAB® R2022b

example 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';
    'elcentro_NS_trunc'; % 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;
```

Set the postyield stiffness factor for each earthquake

```
p=[0.02;0.01;0.02;0.01;0.01;0.01;0.02;0.01;0.01;0.01;0.01];
```

Extract nonlinear response spectra

```
sw='cdrs';
```

Calculation

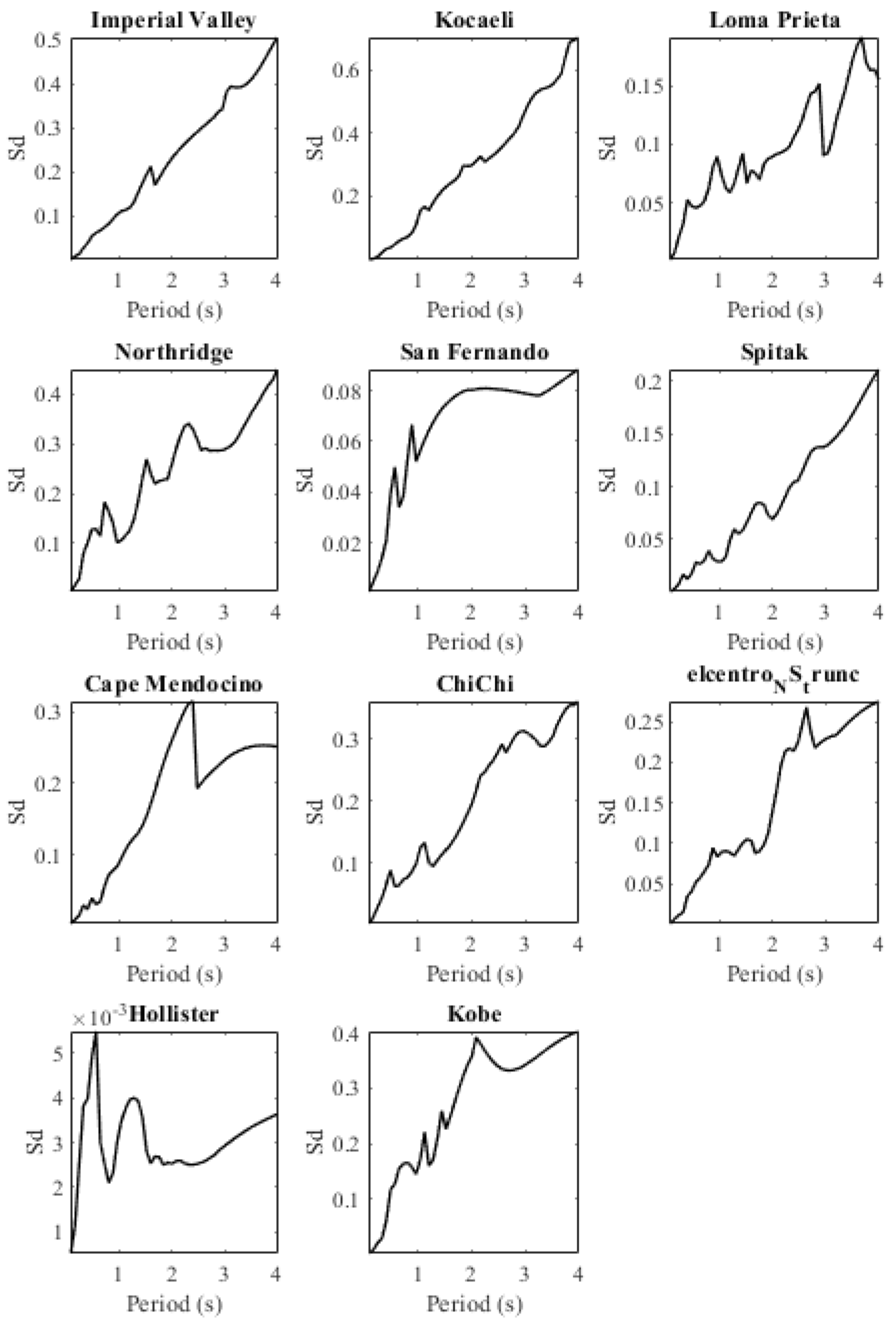
Initialize CDRS

```
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,Tspectra,ksi,mu,p(i));
    CDRS{i}=[S.Period,S.CDSd,S.CDSv,S.CDPSa,S.fyK,S.muK,S.iterK];
end
```

Output

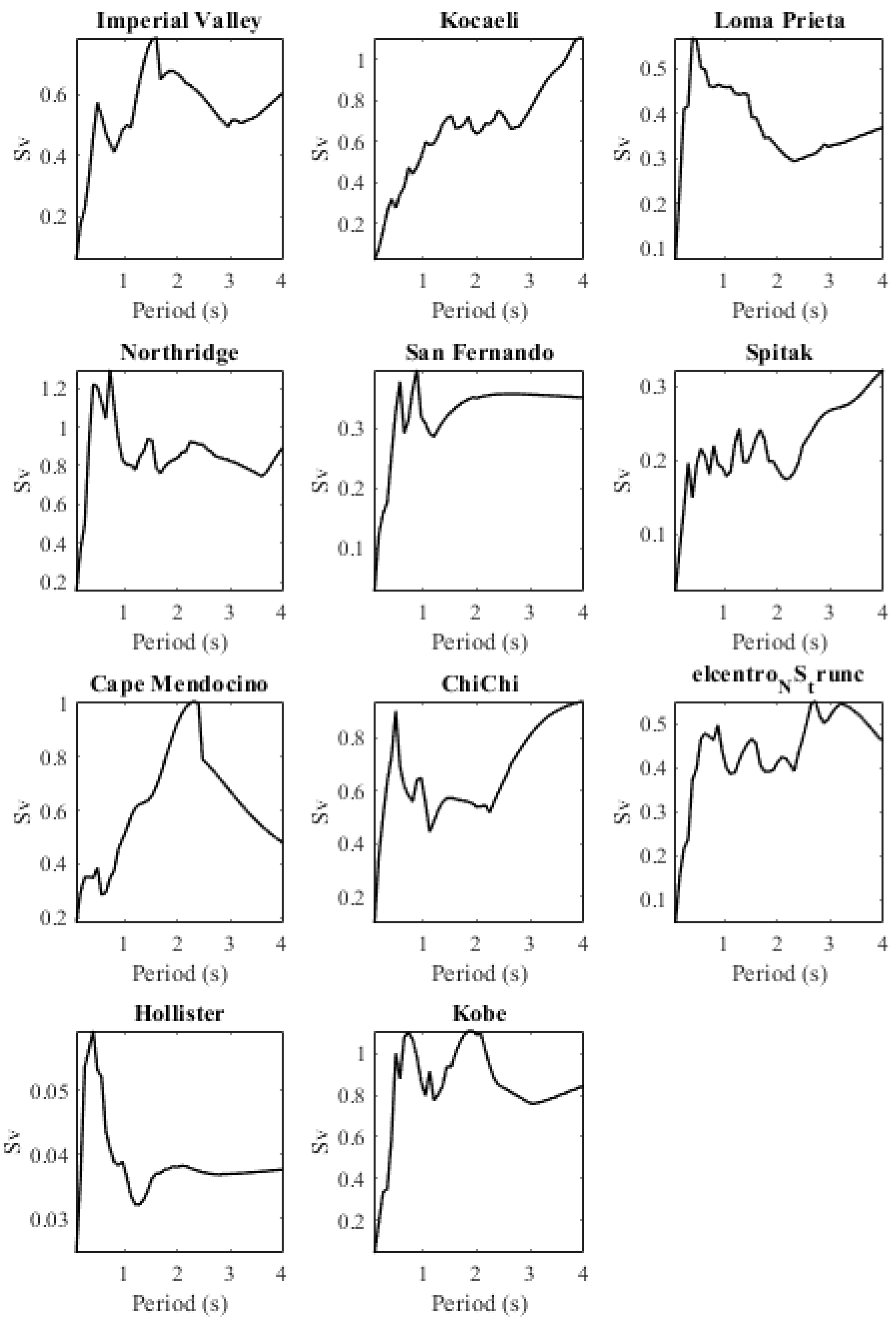
Plot constant ductility spectral displacement

```
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
drawnow;
pause(0.1)
```



Plot constant ductility spectral velocity

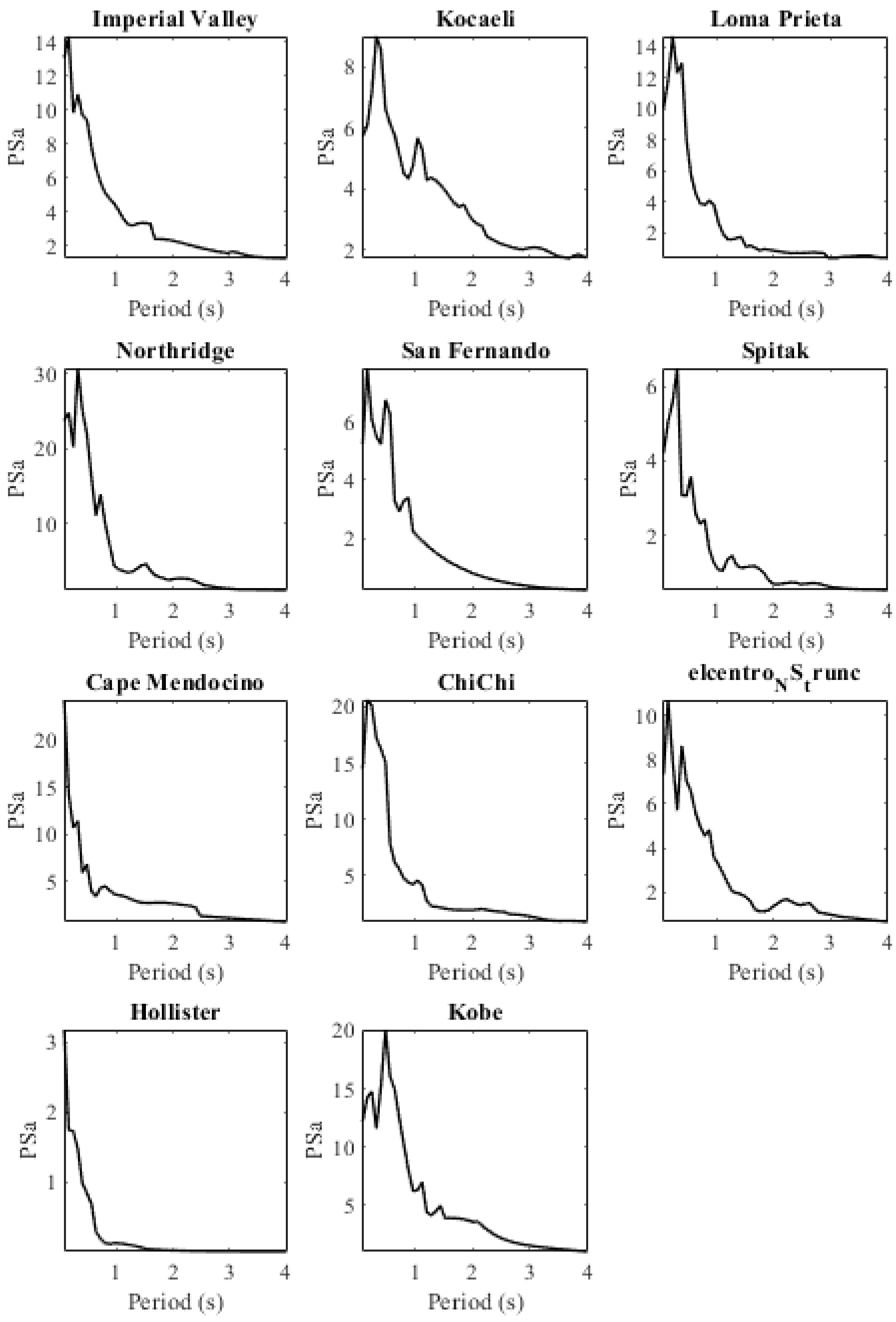
```
Fig2 = figure('units', 'centimeters', 'Position', [0,0, 20/sqrt(2), 20]);
% Scan all subplots
for i=1:numel(eqmotions)
    subplot(4,3,i)
    plot(CDRS{i}(:,1),CDRS{i}(:,3),'k','LineWidth',1);
    set(gca,'FontName','Times New Roman')
    title(eqmotions{i},'FontName','Times New Roman')
    ylabel('Sv','FontName','Times New Roman')
    xlabel('Period (s)','FontName','Times New Roman')
    axis tight
end
drawnow;
pause(0.1)
```



Plot constant ductility spectral acceleration

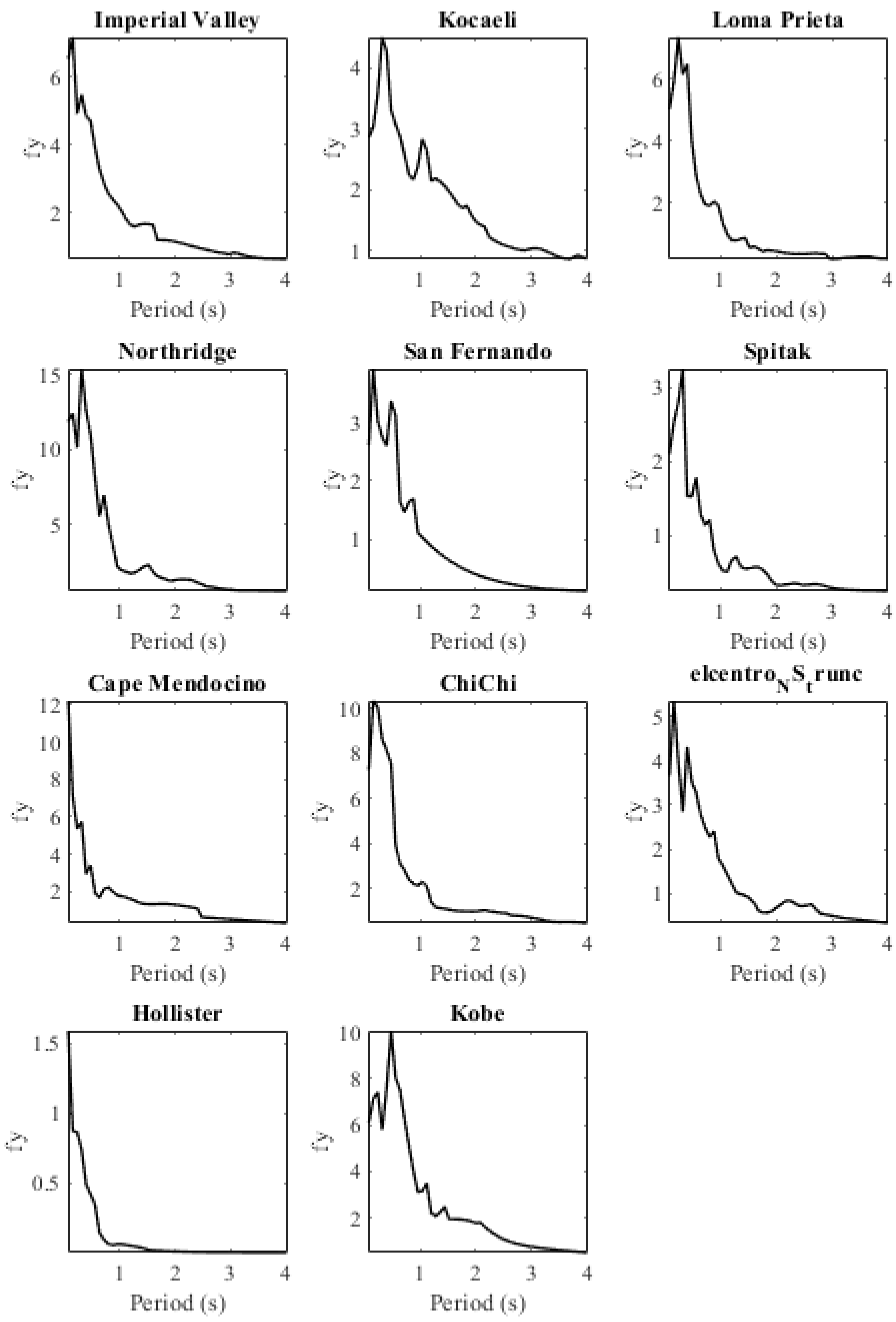
```
Fig3 = figure('units', 'centimeters', 'Position', [0,0, 20/sqrt(2), 20]);
% Scan all subplots
for i=1:numel(eqmotions)
```

```
    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
drawnow;
pause(0.1)
```



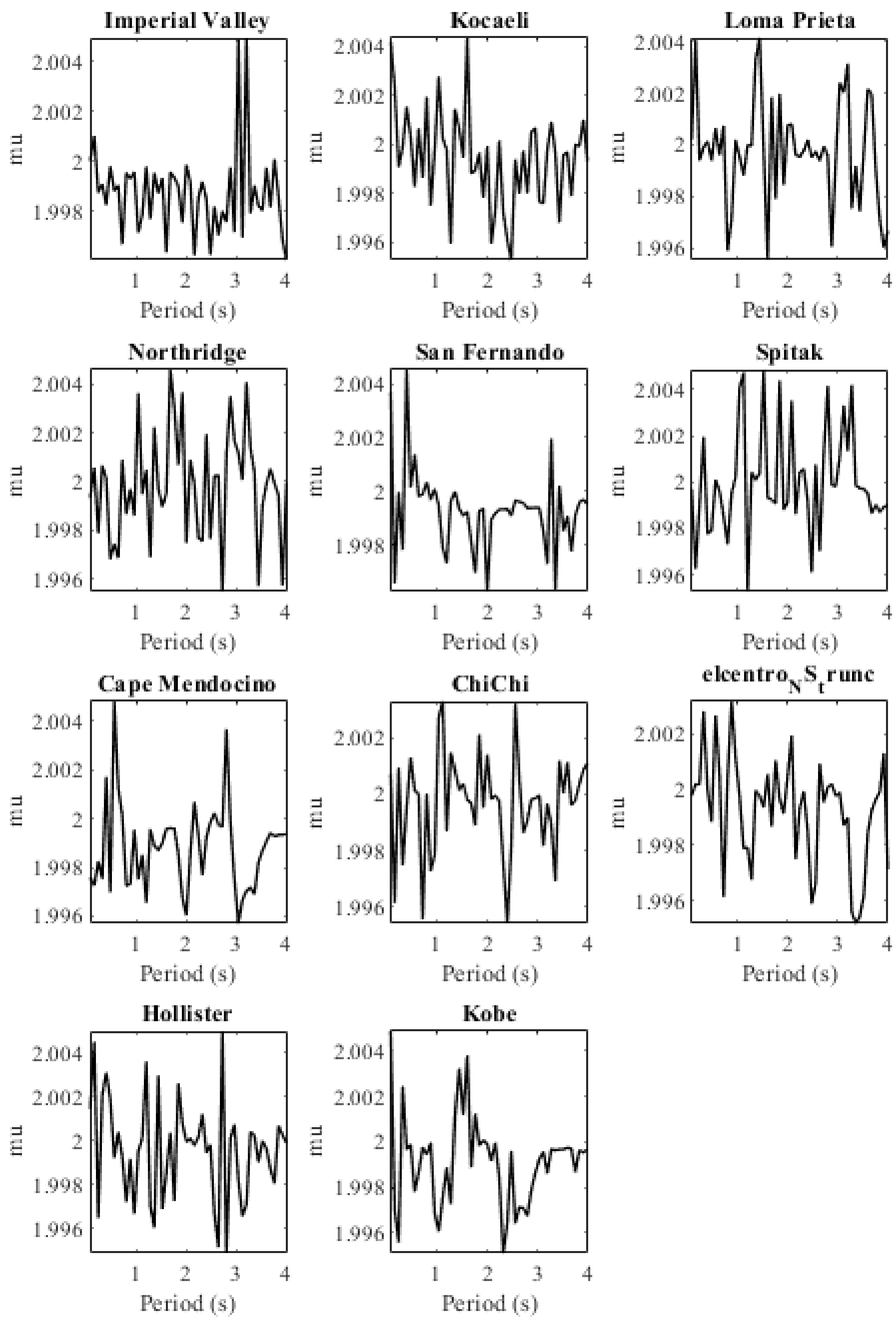
Plot constant ductility spectral yield limit

```
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
drawnow;
pause(0.1)
```



Plot constant ductility spectral achieved ductility

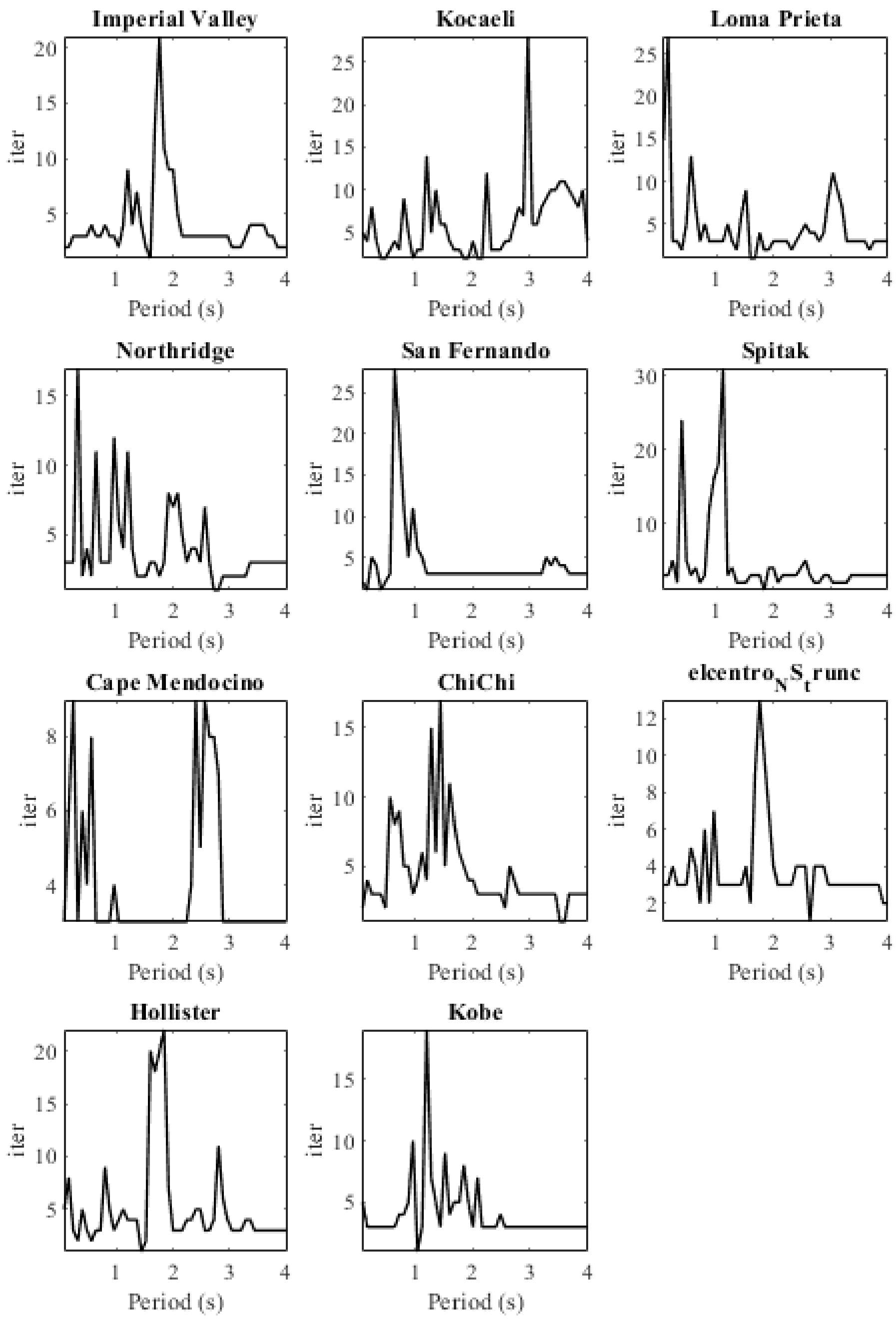
```
Fig5 = figure('units', 'centimeters', 'Position', [0,0, 20/sqrt(2), 20]);
% Scan all subplots
for i=1:numel(eqmotions)
    subplot(4,3,i)
    plot(CDRS{i}(:,1),CDRS{i}(:,6),'k','LineWidth',1);
    set(gca,'FontName','Times New Roman')
    title(eqmotions{i},'FontName','Times New Roman')
    ylabel('mu','FontName','Times New Roman')
    xlabel('Period (s)','FontName','Times New Roman')
    axis tight
end
drawnow;
pause(0.1)
```



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
drawnow;
pause(0.1)
```



Copyright

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

Published with MATLAB® R2022b

example 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';
    'elcentro_NS_trunc'; % Imperial valley 1940
    'Hollister';
    'Kobe'};
```

Switch

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

Extract fourier spectra

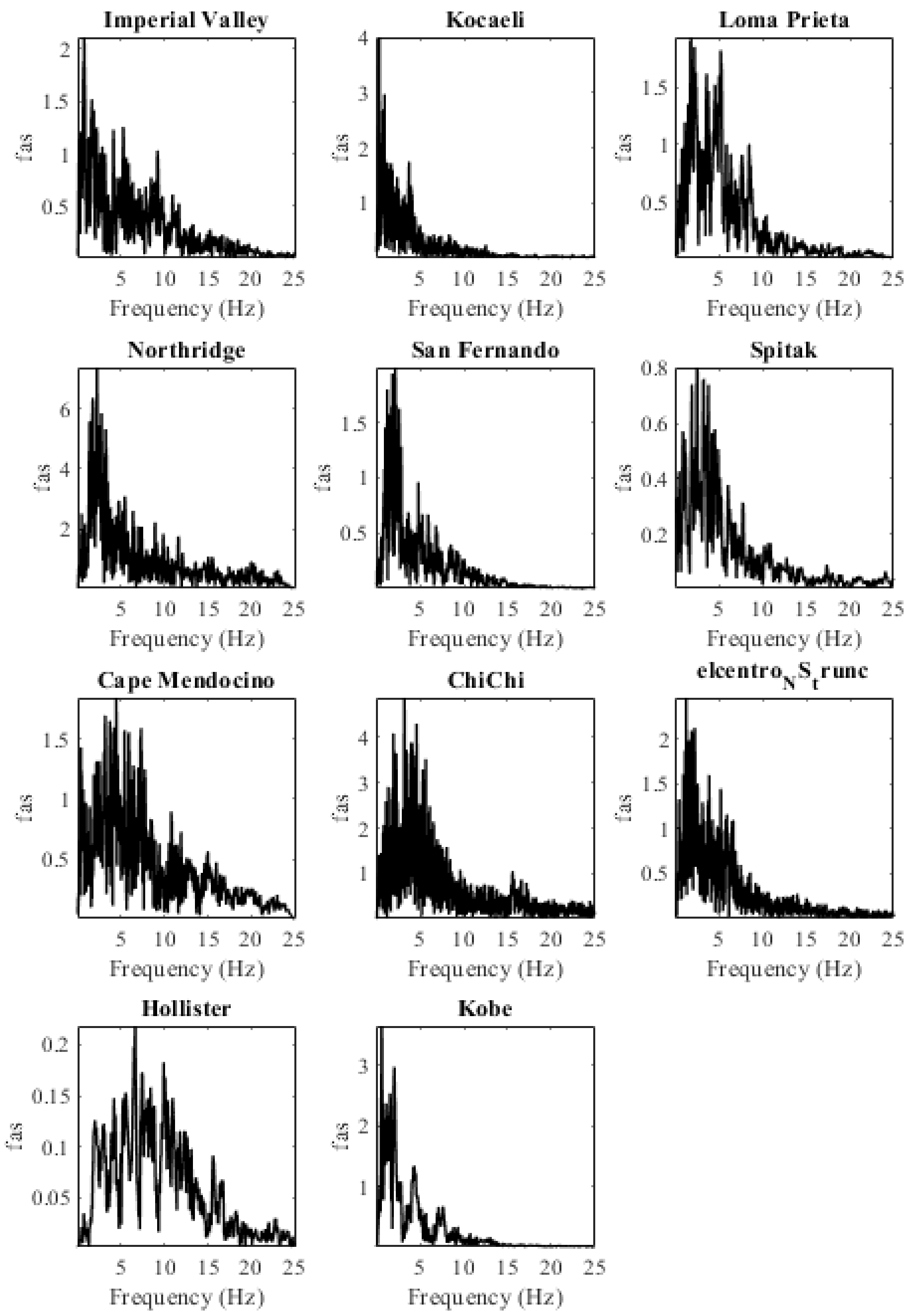
Initialize cell of Fourier spectra

```
Fourier=cell(numel(eqmotions),1);
% Calculation of Fourier spectra
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
drawnow;
pause(0.1)
```



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

Published with MATLAB® R2022b

example General demonstration of the OpenSeismoMatlab capabilities

This is to demonstrate the capabilities of OpenSeismoMatlab and also to verify that OpenSeismoMatlab works properly for all of its possible options and types of application, and yields meaningful results. **All** capabilities of OpenSeismoMatlab are tested in this example

Contents

Load earthquake data

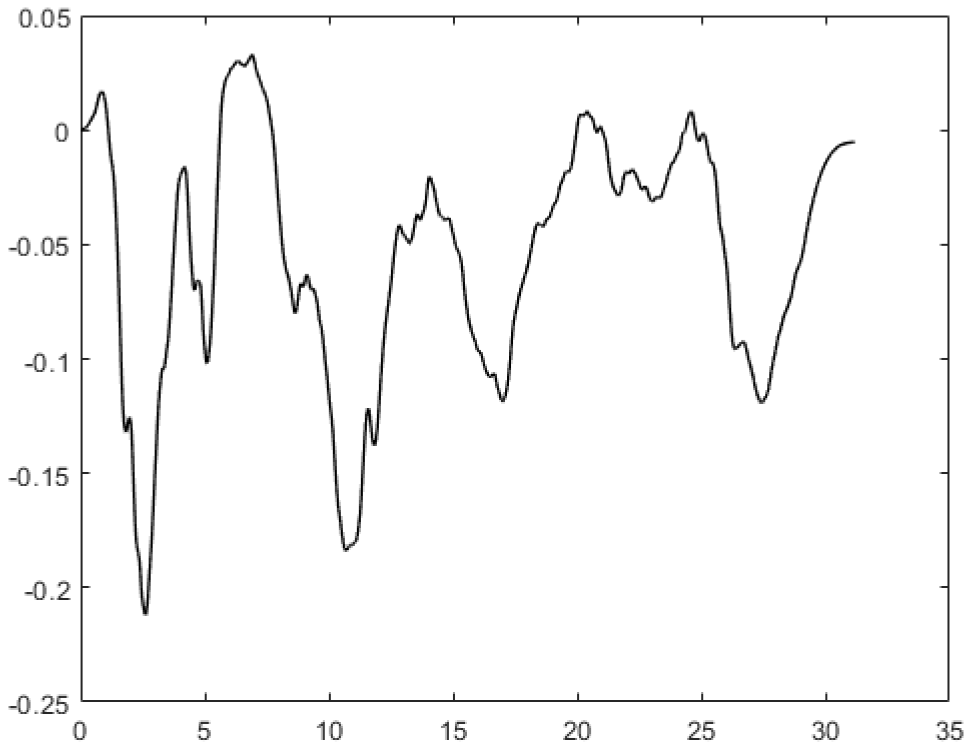
Earthquake acceleration time history of the El Centro earthquake will be used (El Centro, 1940, El Centro Terminal Substation Building)

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

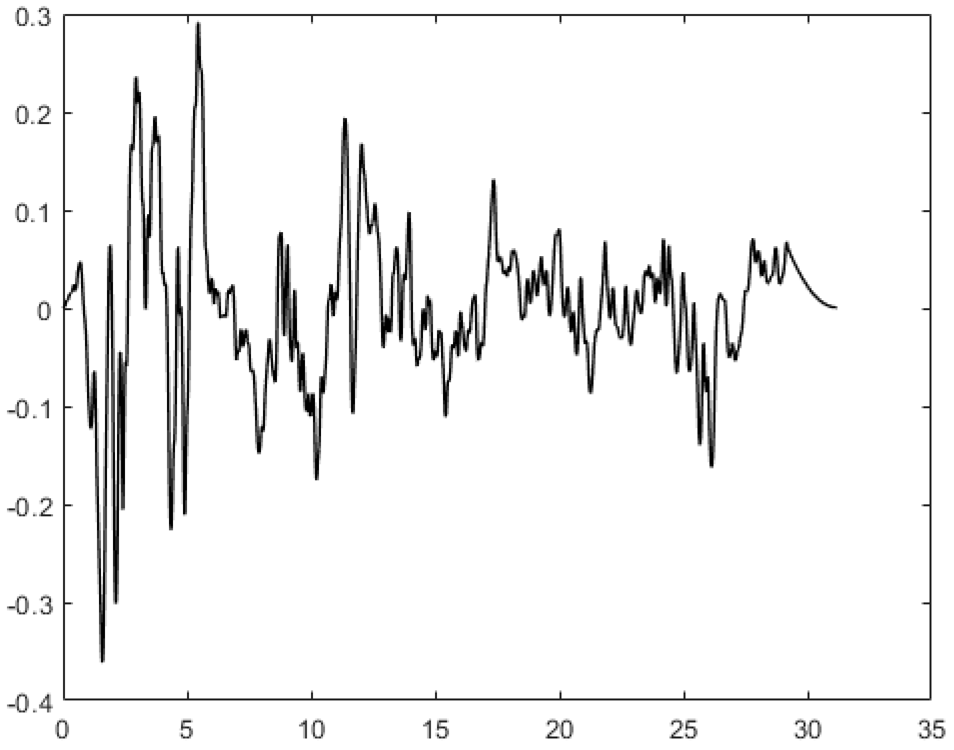
Time histories without baseline correction

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

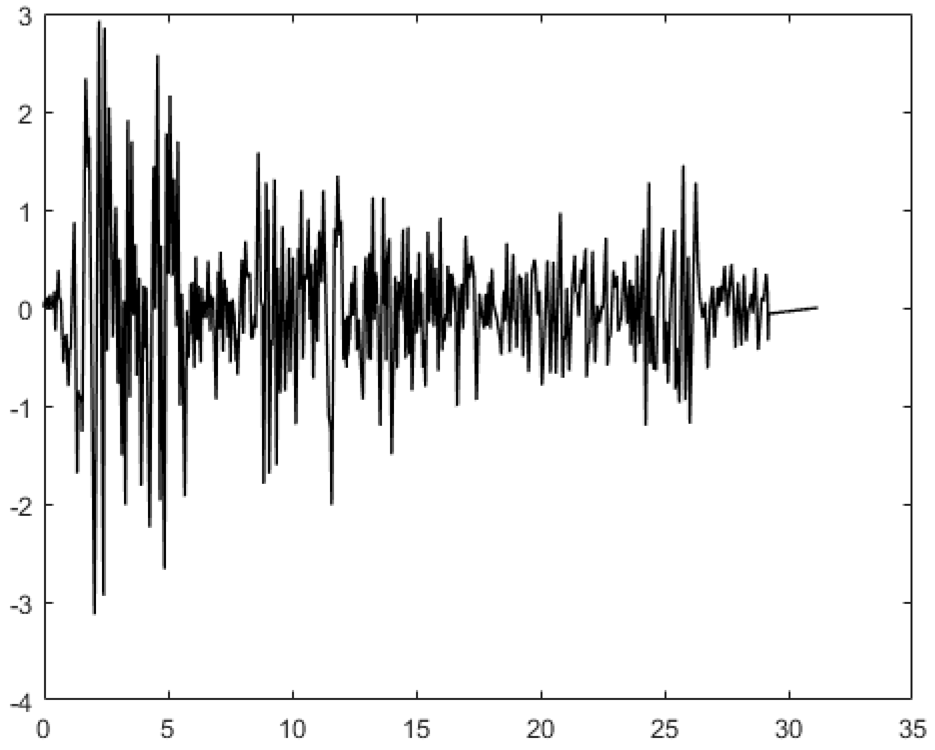
```
figure()
plot(S1.time,S1.disp,'k','LineWidth',1)
drawnow;
pause(0.1)
```



```
figure()
plot(S1.time,S1.vel,'k','LineWidth',1)
drawnow;
pause(0.1)
```



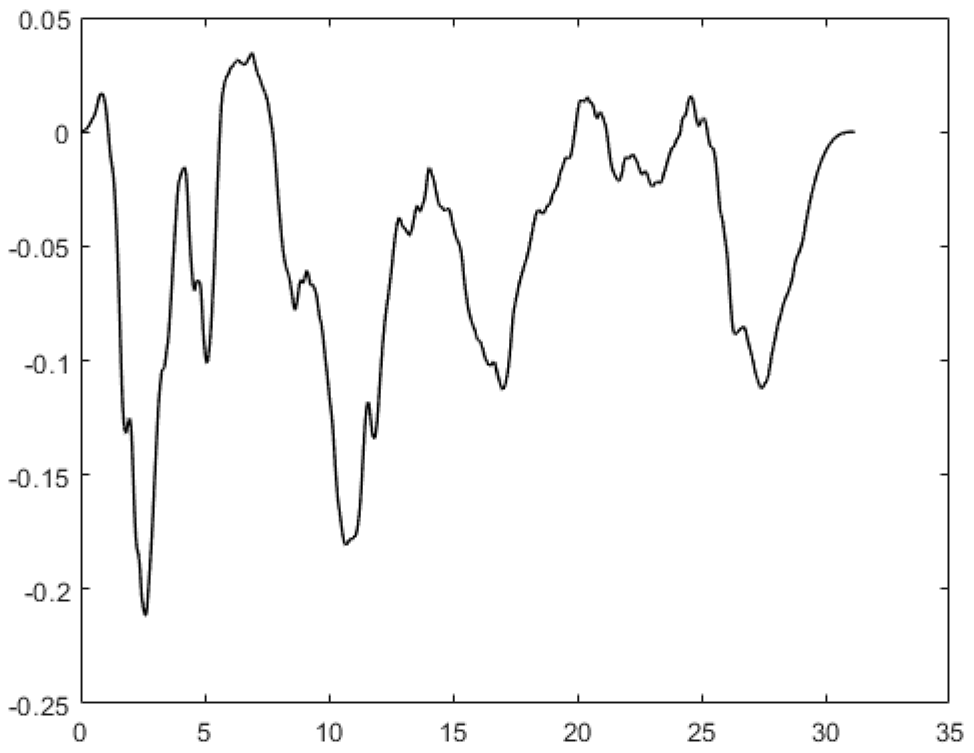
```
figure()
plot(S1.time,S1.acc,'k','LineWidth',1)
drawnow;
pause(0.1)
```



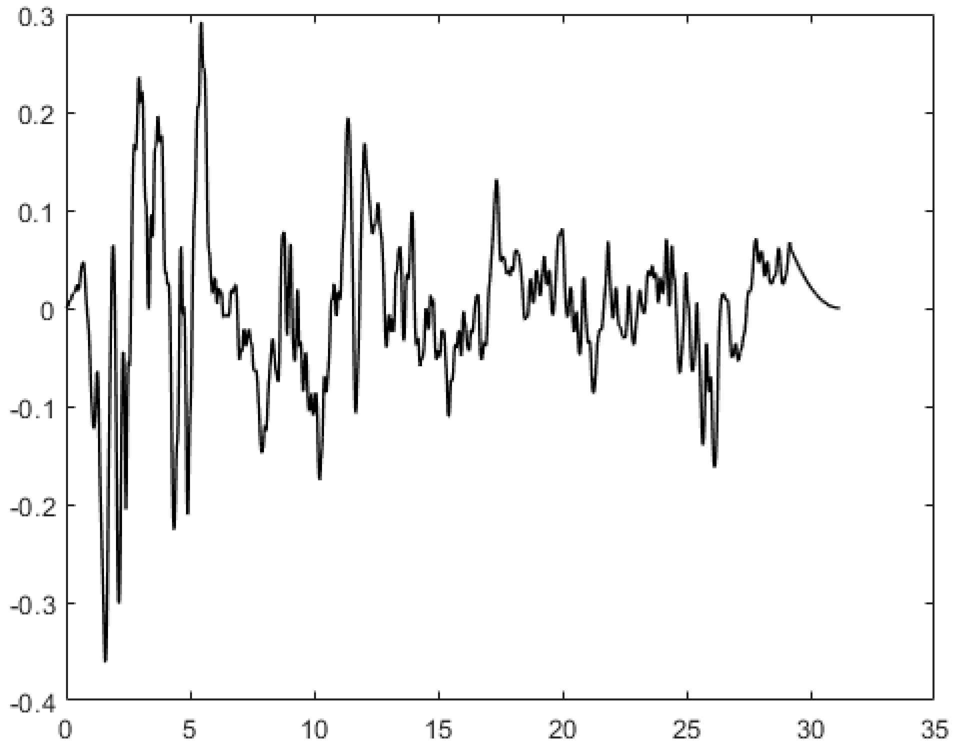
Time histories with baseline correction

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

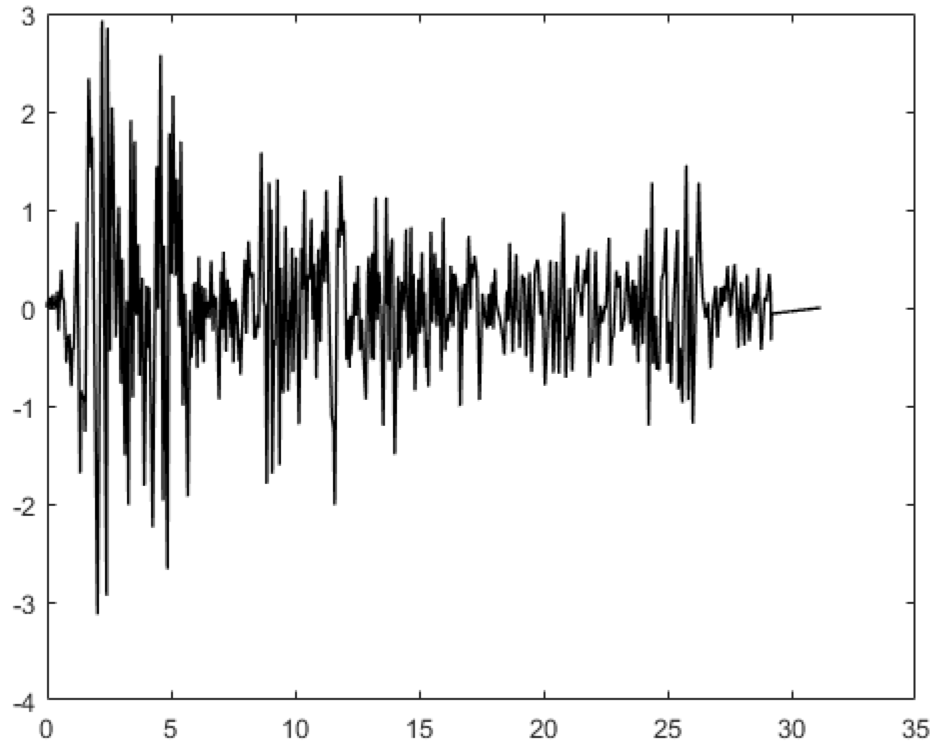
```
figure()
plot(S2.time,S2.disp,'k','LineWidth',1)
drawnow;
pause(0.1)
```



```
figure()
plot(S2.time,S2.vel,'k','LineWidth',1)
drawnow;
pause(0.1)
```



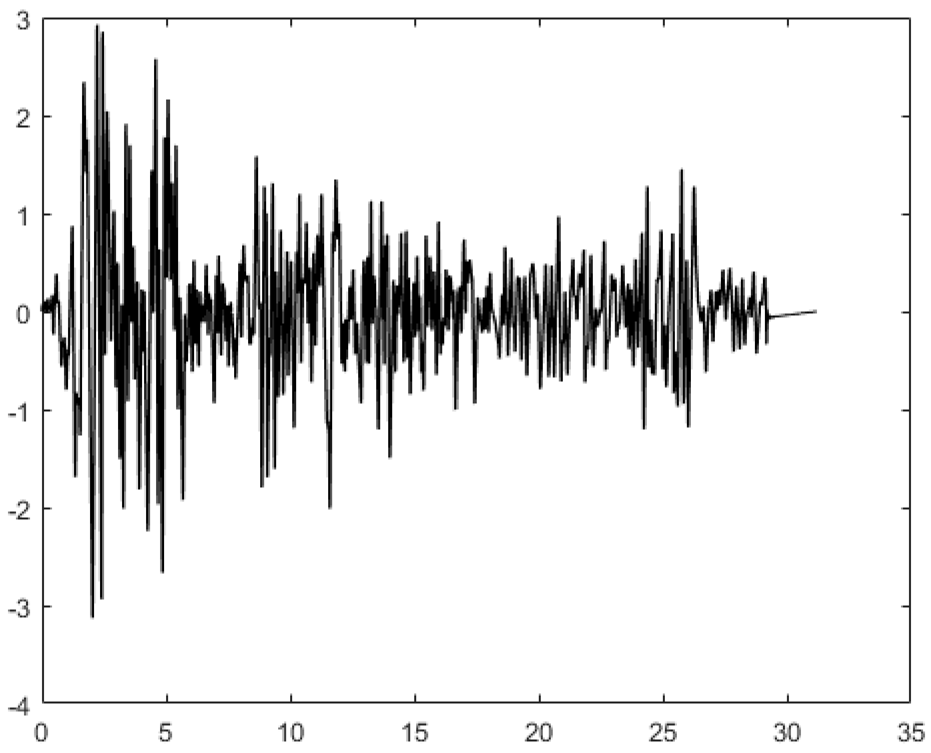
```
figure()
plot(S2.time,S2.acc,'k','LineWidth',1)
drawnow;
pause(0.1)
```



Resample acceleration time history from 0.02 sec to 0.01 sec.

```
sw='sincresample';
dti=0.01;
S3=OpenSeismoMatlab(dt,xgtt,sw,dti);
```

```
figure()
plot(S3.time,S3.acc,'k','LineWidth',1)
drawnow;
pause(0.1)
```



PGA

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

```
S4.PGA
```

```
ans =

                 3.1276242
```

PGV

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

```
S5.PGV
```

```
ans =

                 0.360920691
```

PGD

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

```
S6.PGD
```

```
ans =

          0.211893410160001
```

Total cumulative energy, Arias intensity and significant duration

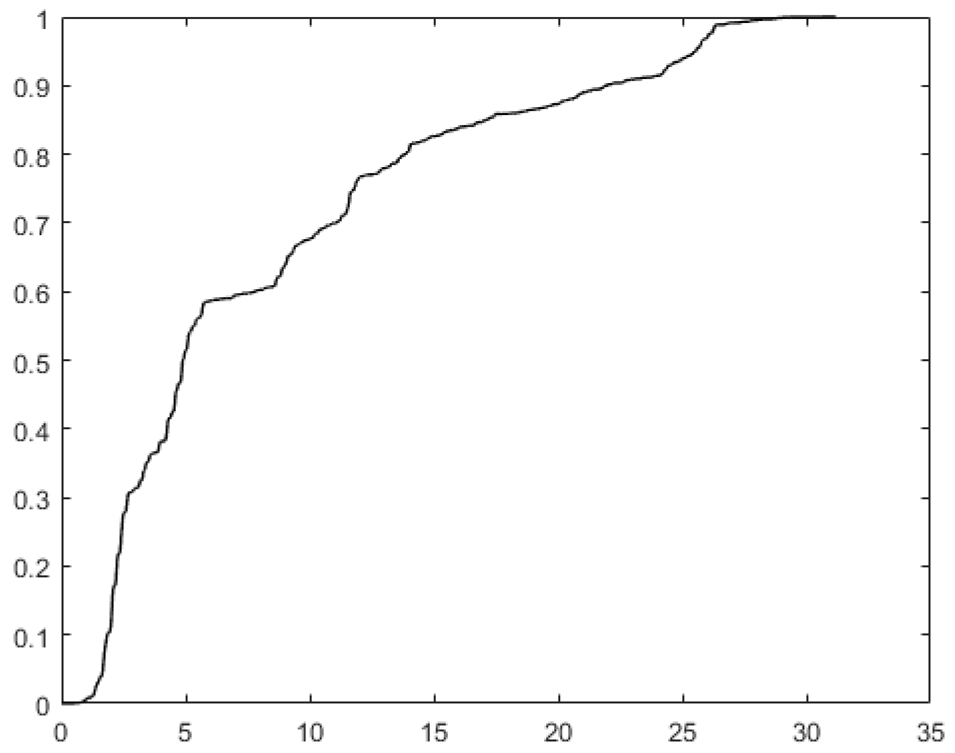
```
sw='arias';
S7=OpenSeismoMatlab(dt,xgtt,sw);
```

```
S7.Ecum
```

```
ans =

            11.251388628142
```

```
figure()
plot(S7.time,S7.EcumTH,'k','LineWidth',1)
drawnow;
pause(0.1)
```



```
S7.t_5_95
```

```
ans =

                      1.68                      25.5
```

```
S7.Td_5_95
```

```
ans =

                     23.84
```

```
S7.t_5_75
```

```
ans =

                      1.68                      11.8
```

```
S7.Td_5_75
```

```
ans =

                     10.14
```

```
S7.arias
```

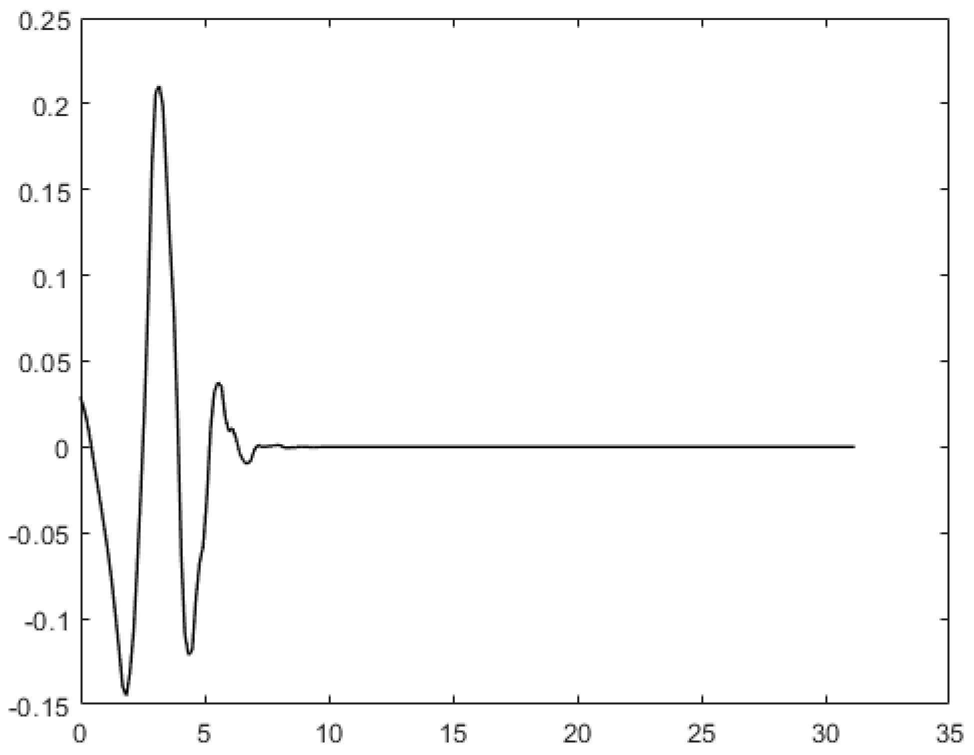
```
ans =

          1.80159428424335
```

Pulse decomposition

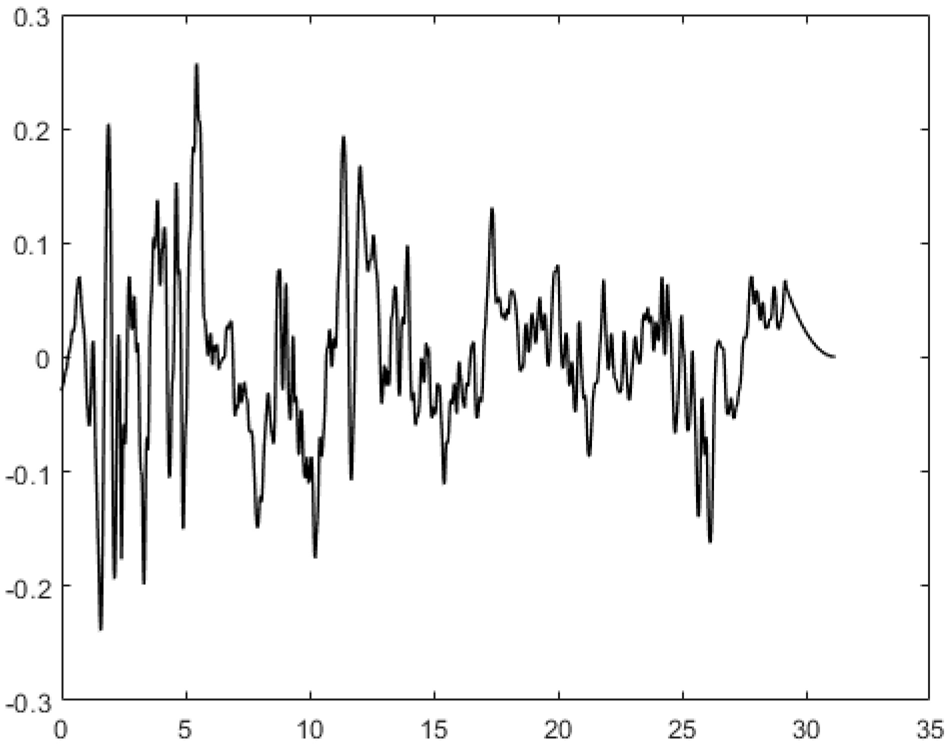
```
xgt=S1.vel;
sw='pulsedecomp';
S8=OpenSeismoMatlab(dt,xgt,sw);
```

```
figure()
plot(S8.time,S8.pulseTH,'k','LineWidth',1)
drawnow;
pause(0.1)
```



```
figure()
```

```
plot(S8.time,S8.resTH,'k','LineWidth',1)
drawnow;
pause(0.1)
```



```
S8.Tp
```

```
ans =

                     3.304
```

```
S8.wavScale
```

```
ans =

   118
```

```
S8.wavCoefs
```

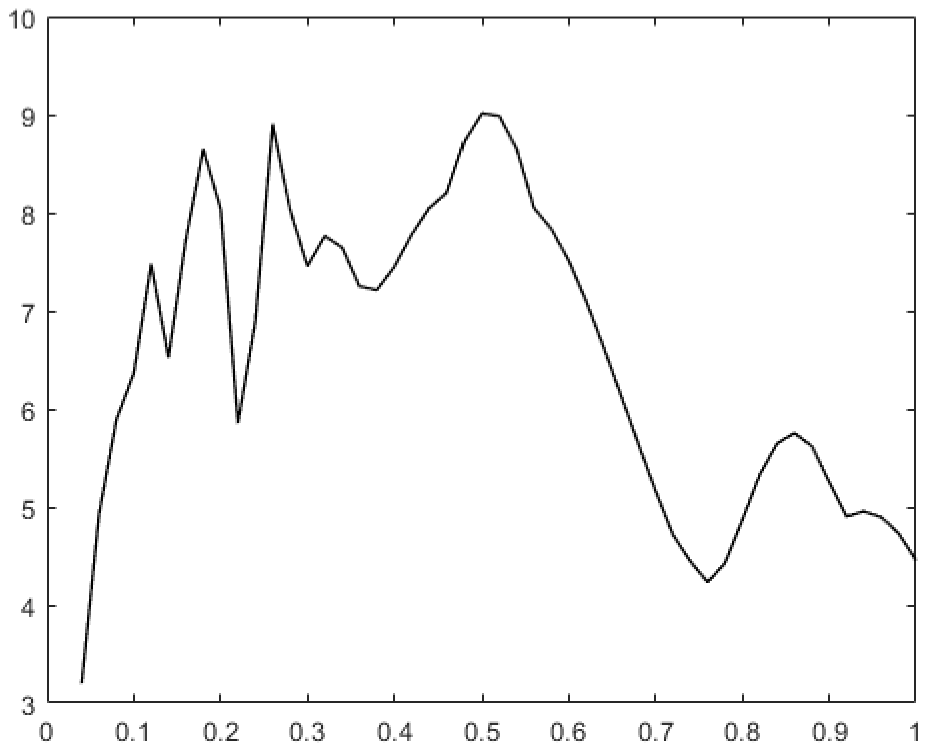
```
ans =

          1.71247132221248
```

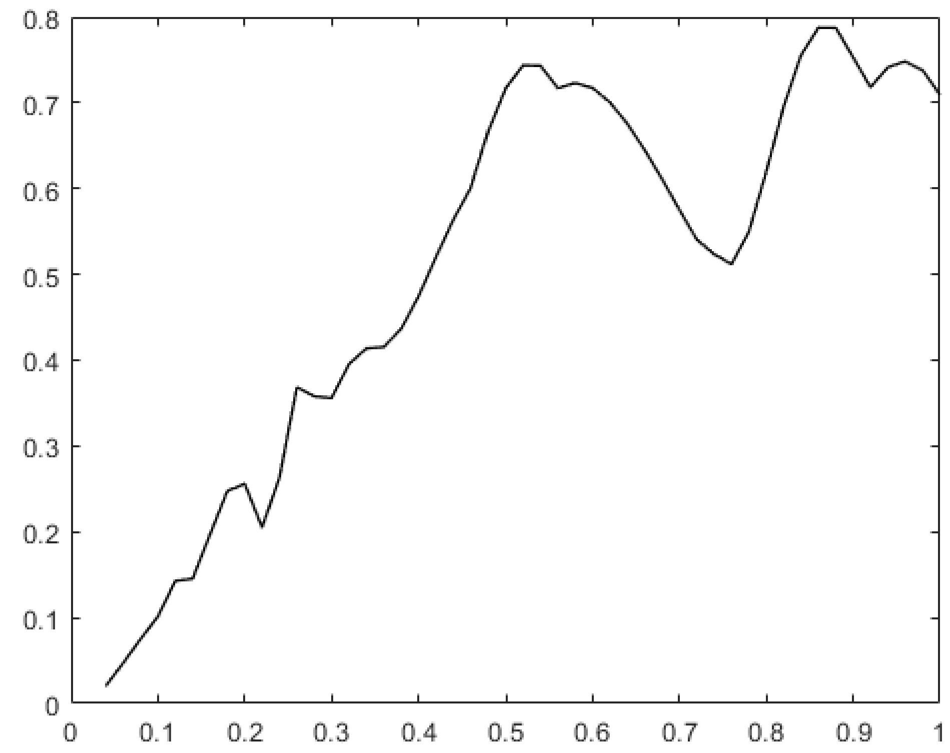
Linear elastic response spectra and pseudospectra

```
sw='elrs';
ksi=0.05;
T=0.04:0.02:1;
S9=OpenSeismoMatlab(dt,xgtt,sw,T,ksi);
```

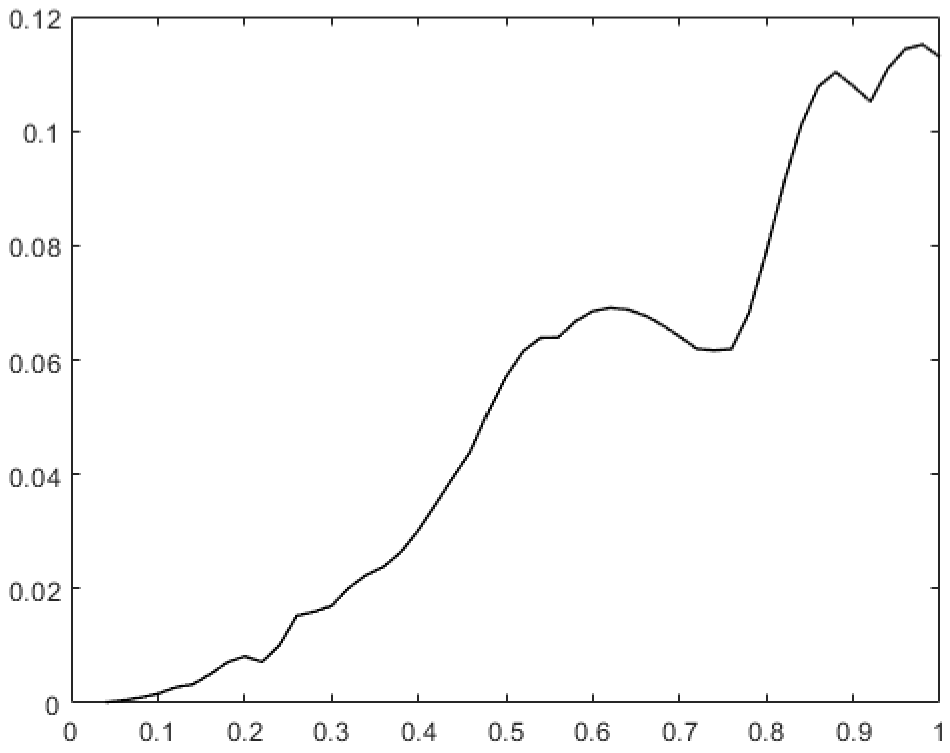
```
figure()
plot(S9.Period,S9.PSa,'k','LineWidth',1)
drawnow;
pause(0.1)
```



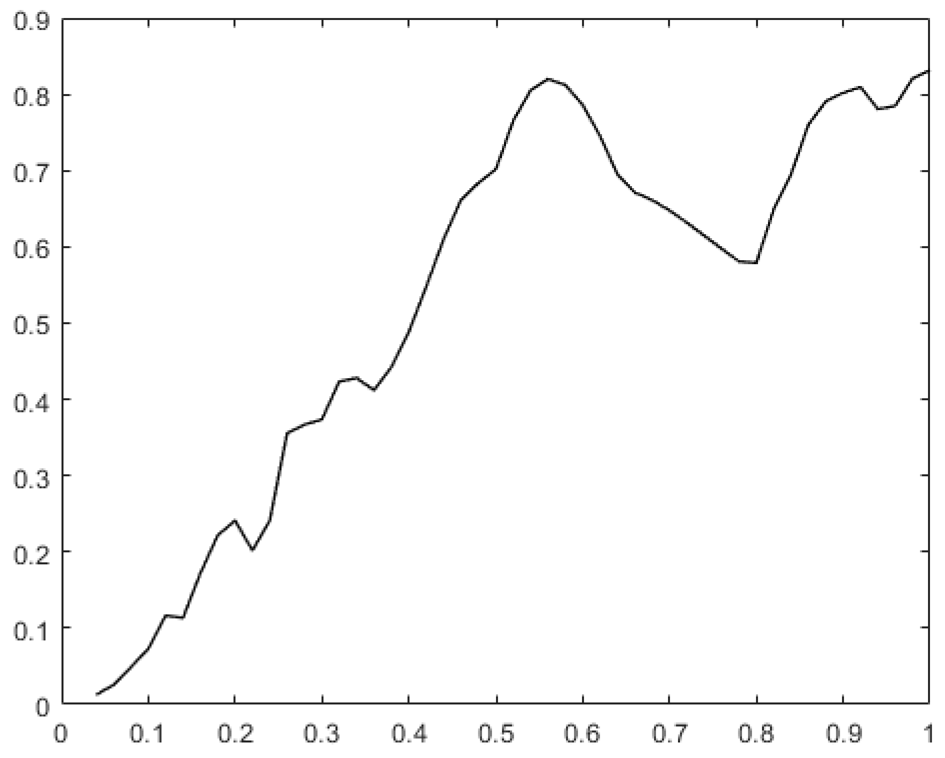
```
figure()
plot(S9.Period,S9.PSv,'k','LineWidth',1)
drawnow;
pause(0.1)
```



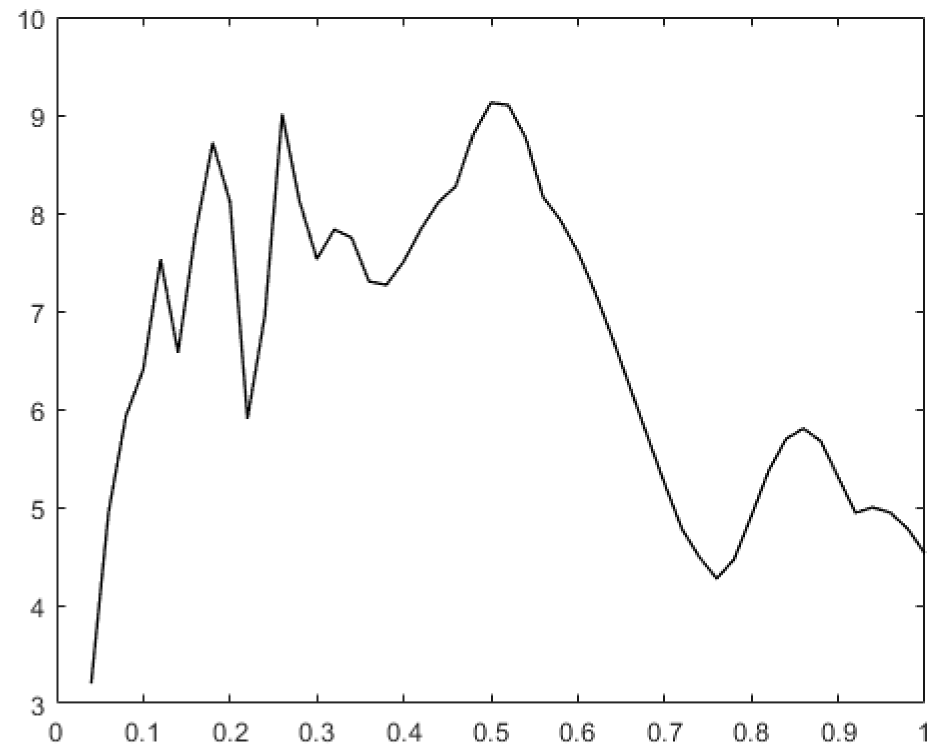
```
figure()
plot(S9.Period,S9.Sd,'k','LineWidth',1)
drawnow;
pause(0.1)
```



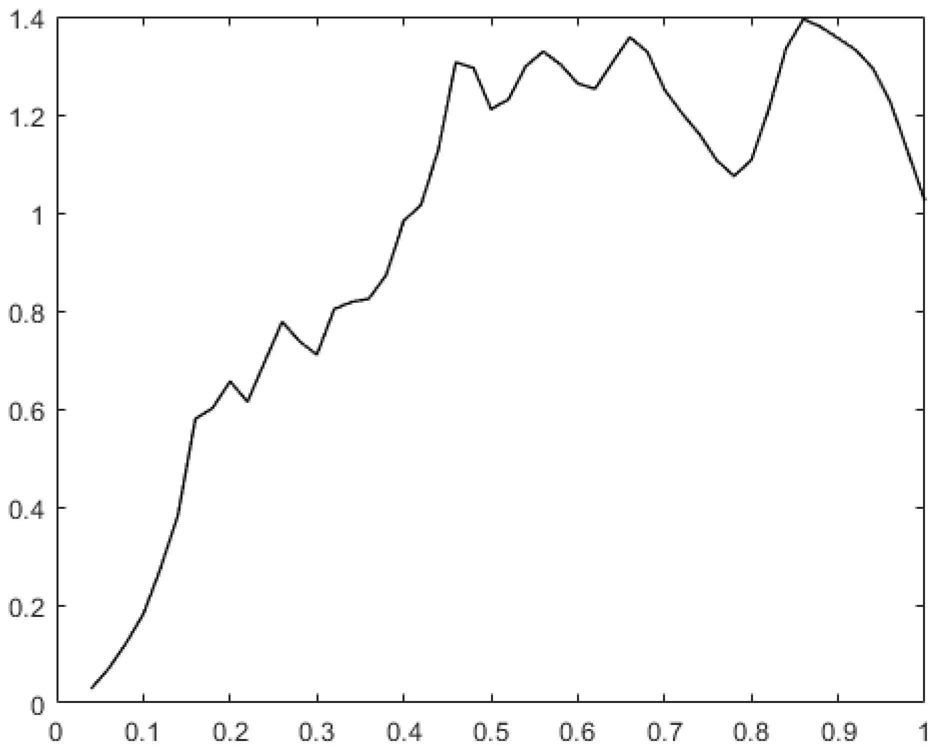
```
figure()
plot(S9.Period,S9.Sv,'k','LineWidth',1)
drawnow;
pause(0.1)
```



```
figure()
plot(S9.Period,S9.Sa,'k','LineWidth',1)
drawnow;
pause(0.1)
```



```
figure()
plot(S9.Period,S9.Siev,'k','LineWidth',1)
drawnow;
pause(0.1)
```



```
S9.PredPSa
```

```
ans =

          9.01181525615902
```

```
S9.PredPeriod
```

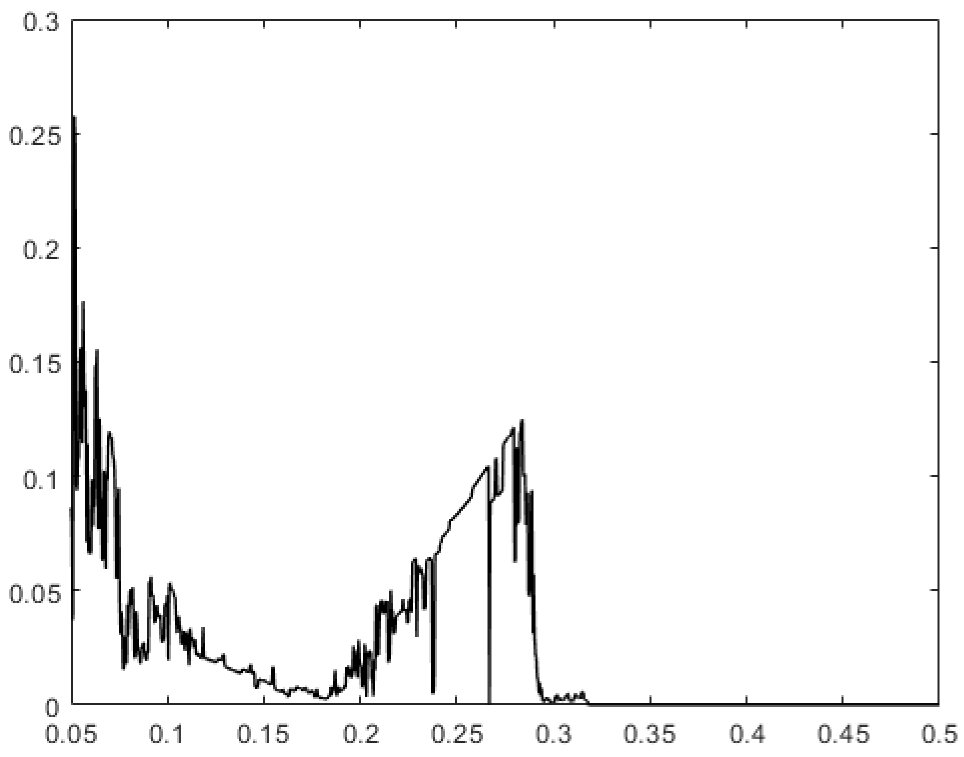
```
ans =

                       0.5
```

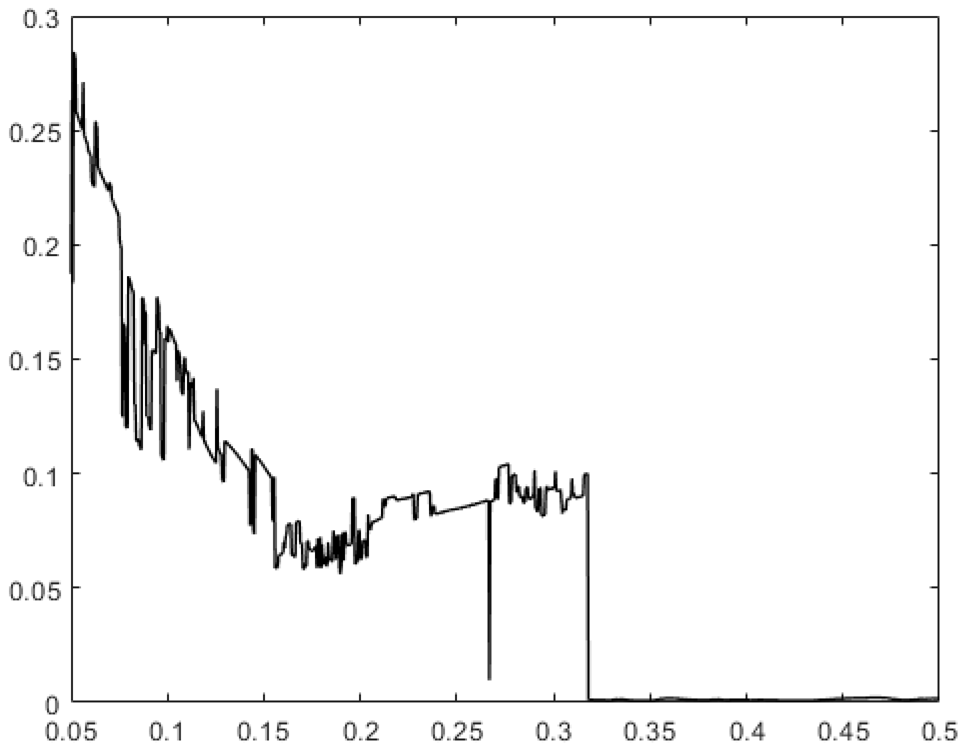
Rigid plastic sliding displacement response spectra

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

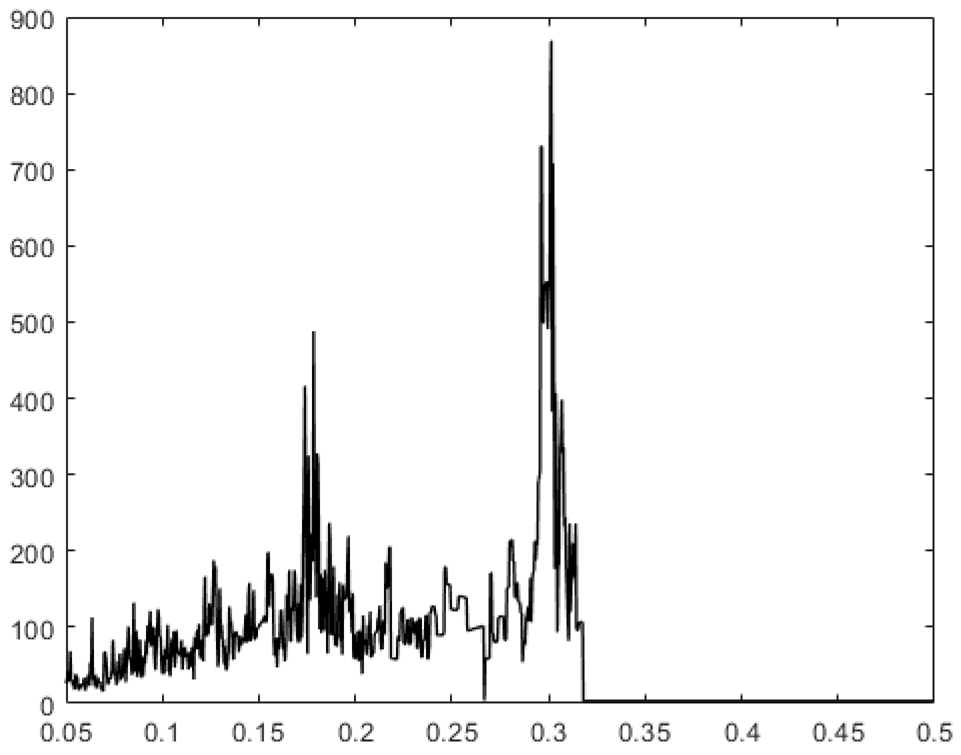
```
figure()
plot(S10.CF,S10.RPSSd,'k','LineWidth',1)
drawnow;
pause(0.1)
```



```
figure()
plot(S10.CF,S10.RPSSv,'k','LineWidth',1)
drawnow;
pause(0.1)
```



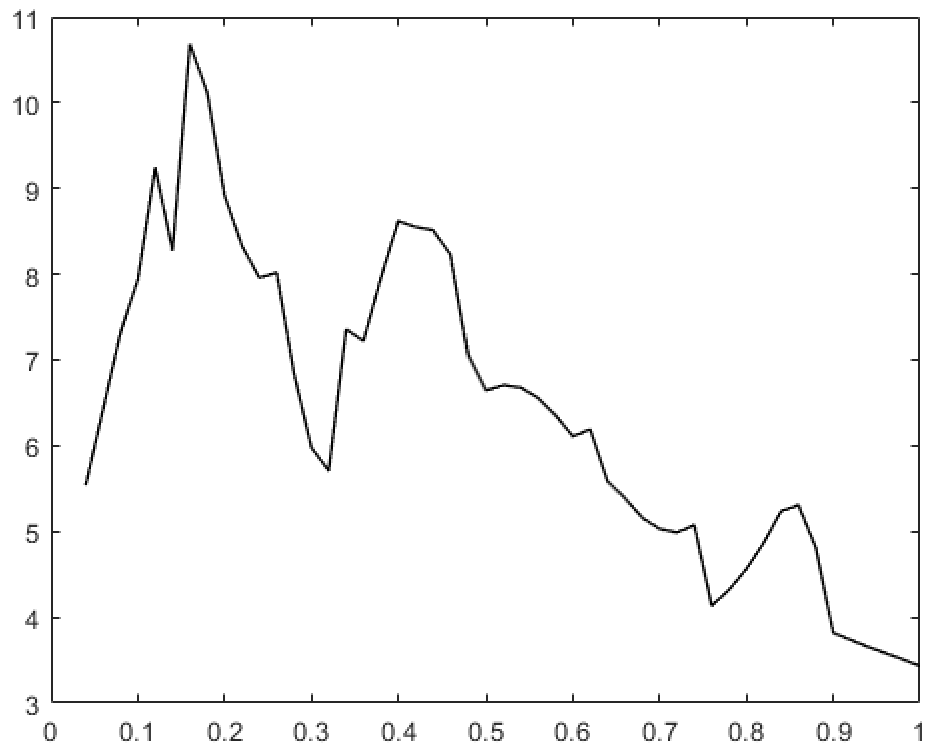
```
figure()
plot(S10.CF,S10.RPSSa,'k','LineWidth',1)
drawnow;
pause(0.1)
```



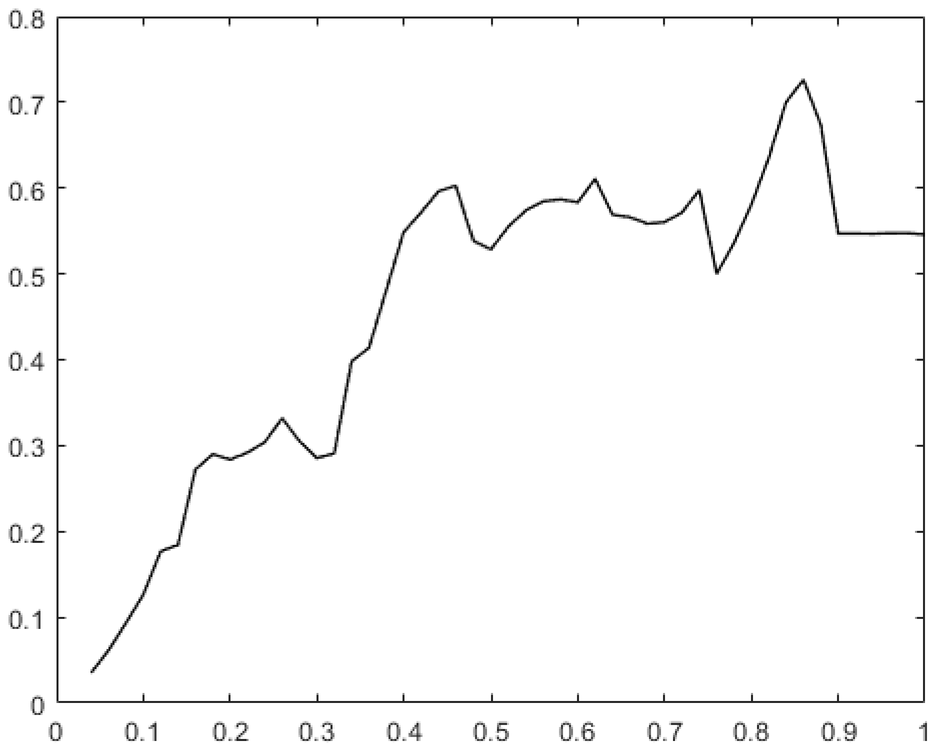
Constant ductility response spectra and pseudospectra

```
sw='cdrs';
ksi=0.05;
T=0.04:0.02:1;
mu=2;
S11=OpenSeismoMatlab(dt,xgtt,sw,T,ksi,mu);
```

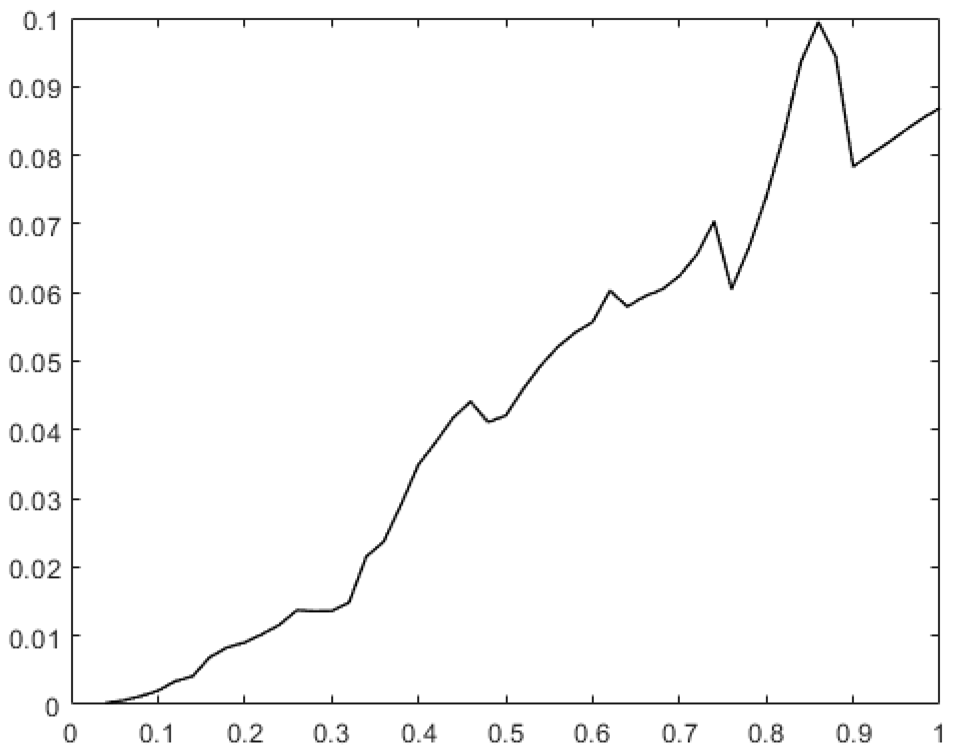
```
figure()
plot(S11.Period,S11.CDPSa,'k','LineWidth',1)
drawnow;
pause(0.1)
```



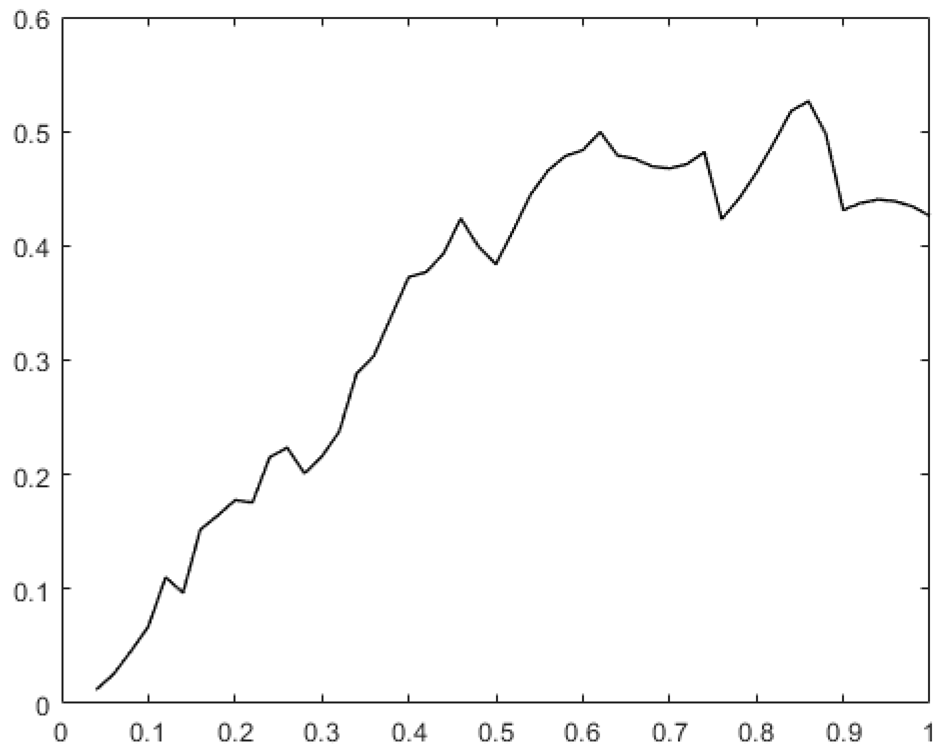
```
figure()
plot(S11.Period,S11.CDPSv,'k','LineWidth',1)
drawnow;
pause(0.1)
```



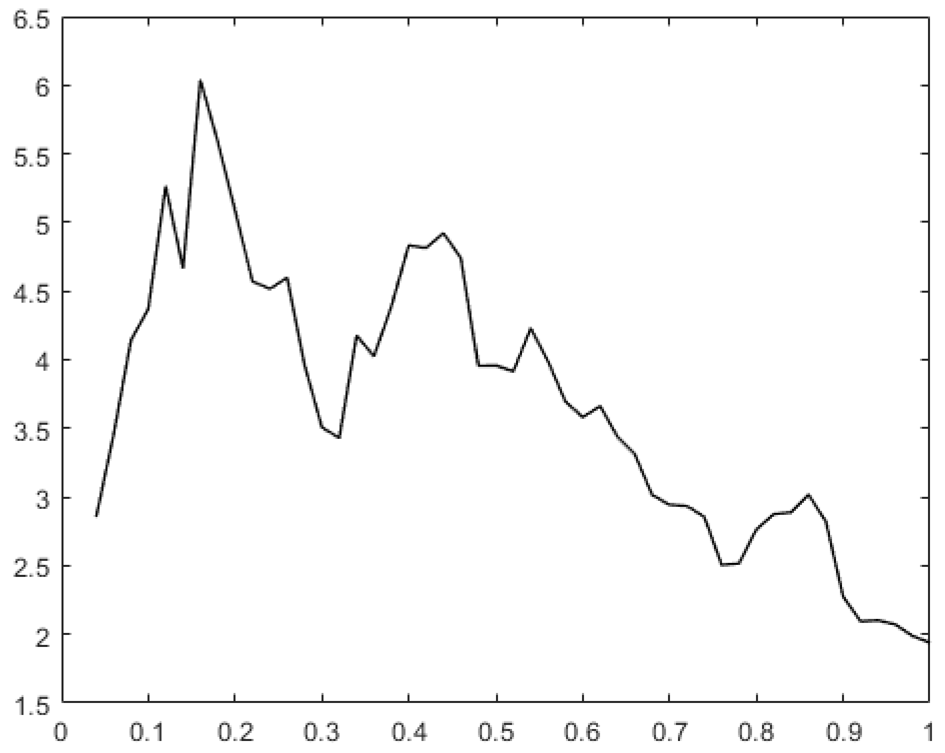
```
figure()
plot(S11.Period,S11.CDSd,'k','LineWidth',1)
drawnow;
pause(0.1)
```



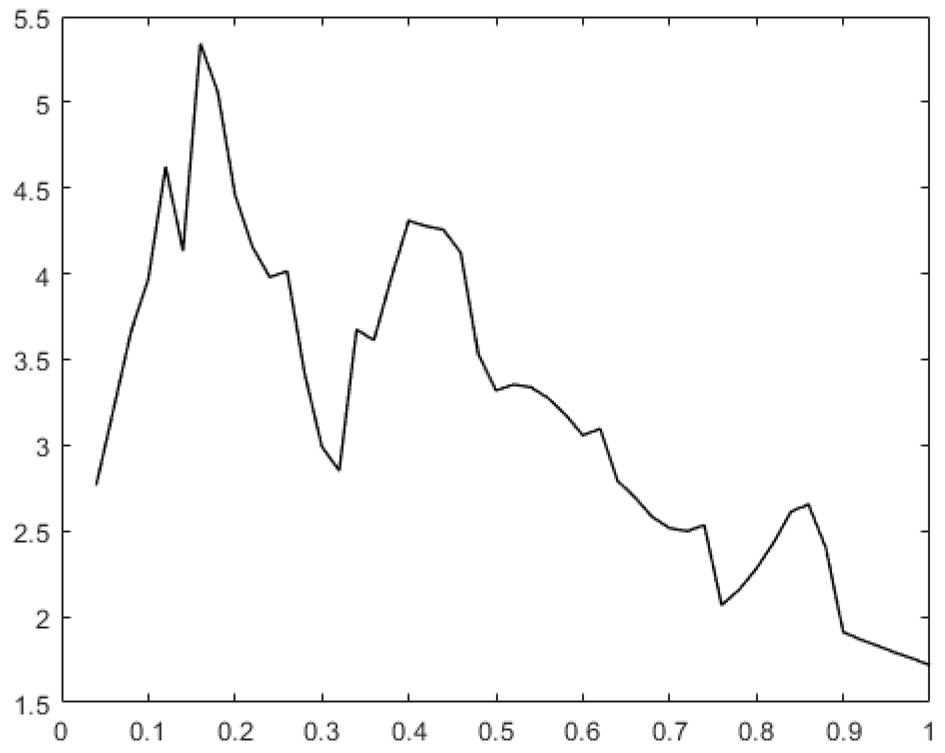
```
figure()
plot(S11.Period,S11.CDSv,'k','LineWidth',1)
drawnow;
pause(0.1)
```



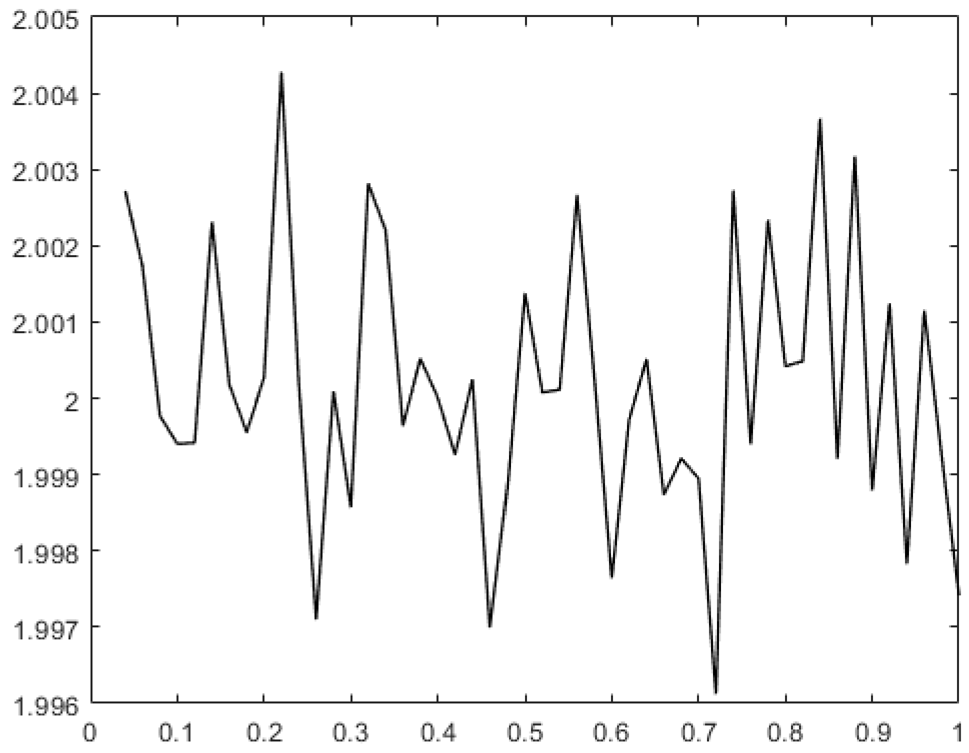
```
figure()
plot(S11.Period,S11.CDSa,'k','LineWidth',1)
drawnow;
pause(0.1)
```



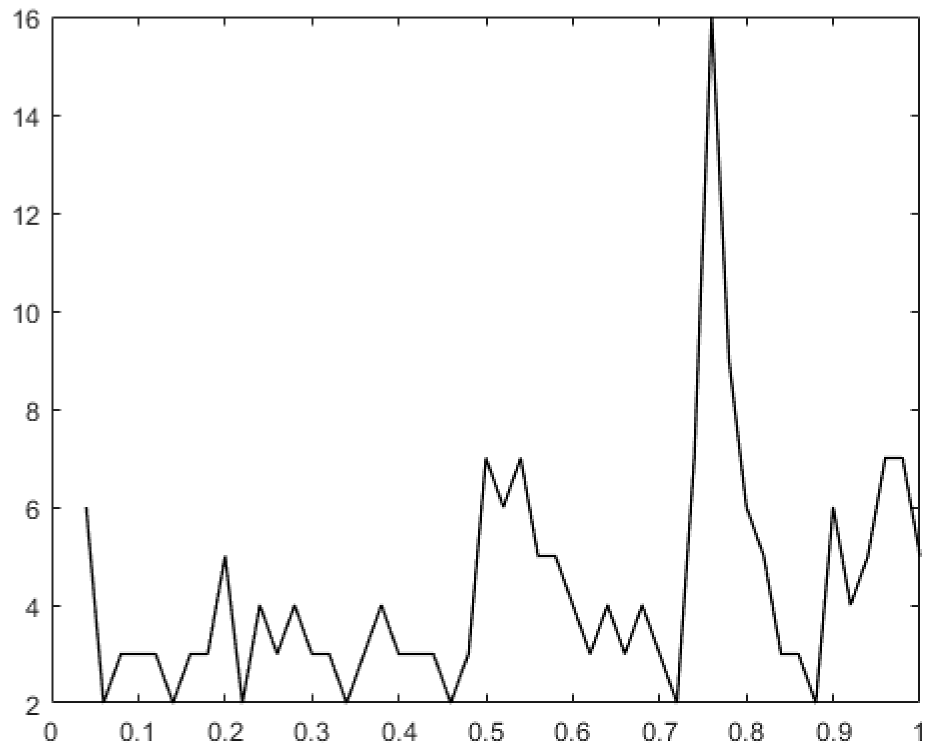
```
figure()
plot(S11.Period,S11.fyK,'k','LineWidth',1)
drawnow;
pause(0.1)
```



```
figure()
plot(S11.Period,S11.muK,'k','LineWidth',1)
drawnow;
pause(0.1)
```



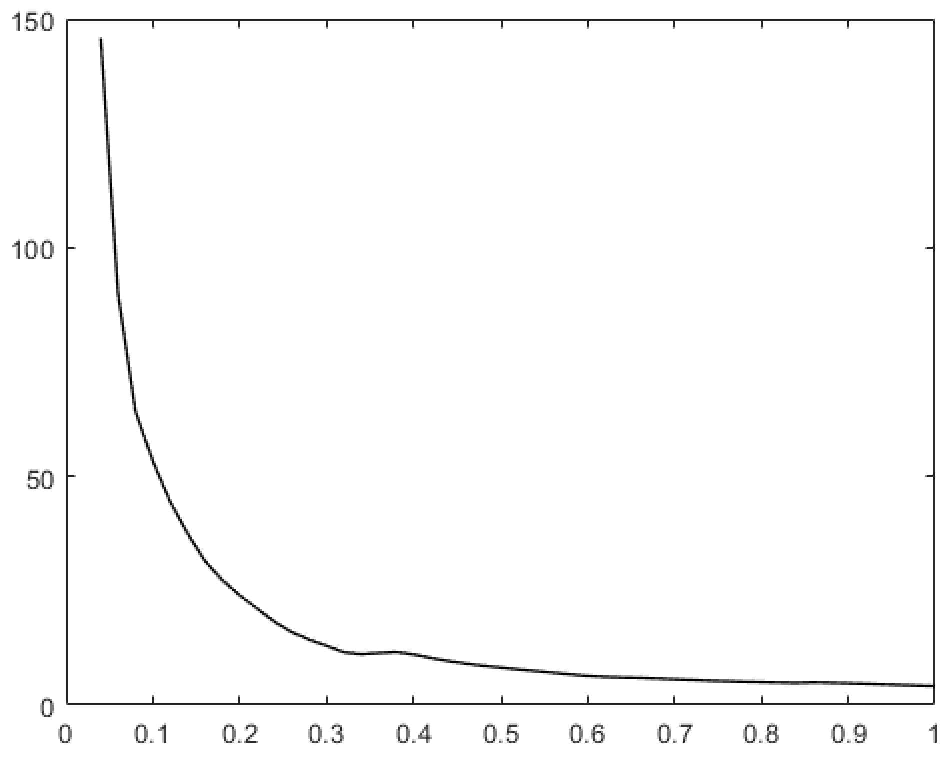
```
figure()
plot(S11.Period,S11.iterK,'k','LineWidth',1)
drawnow;
pause(0.1)
```



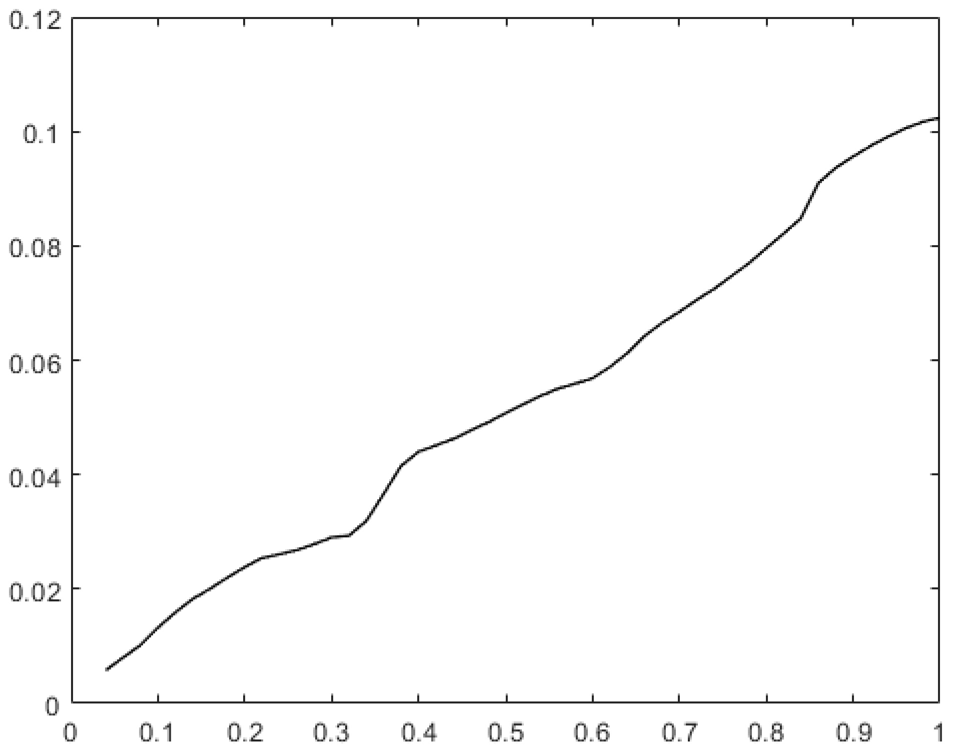
Constant strength response spectra

```
sw='csrs';
ksi=0.05;
T=0.04:0.02:1;
fyR=0.1;
S12=OpenSeismoMatlab(dt,xgtt,sw,T,ksi,fyR);
```

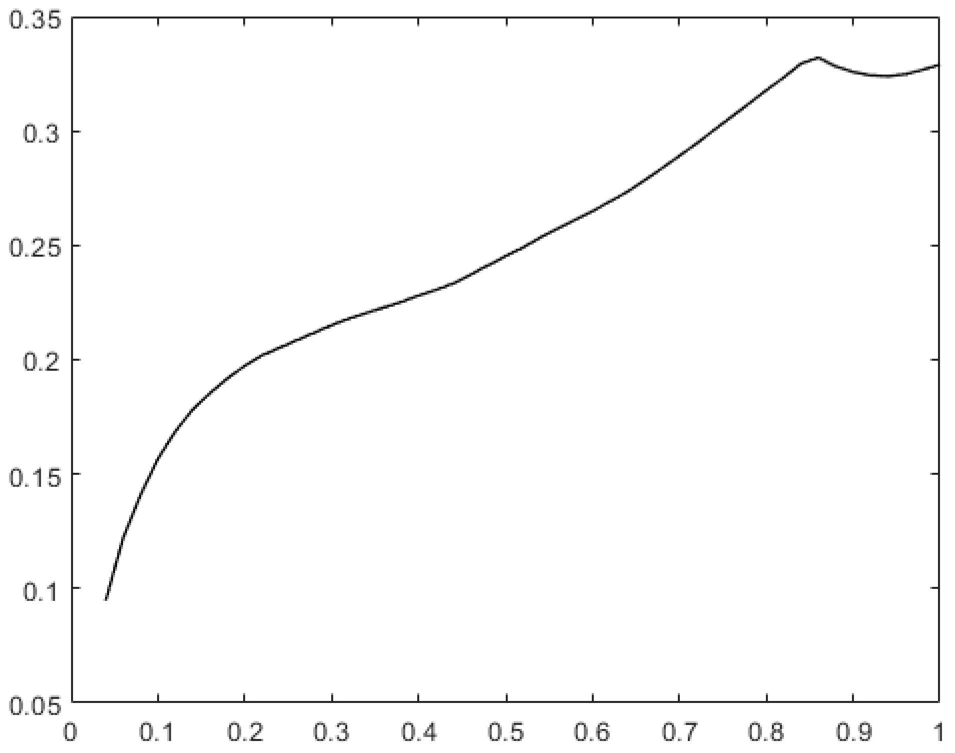
```
figure()
plot(S12.Period,S12.CSSmu,'k','LineWidth',1)
drawnow;
pause(0.1)
```



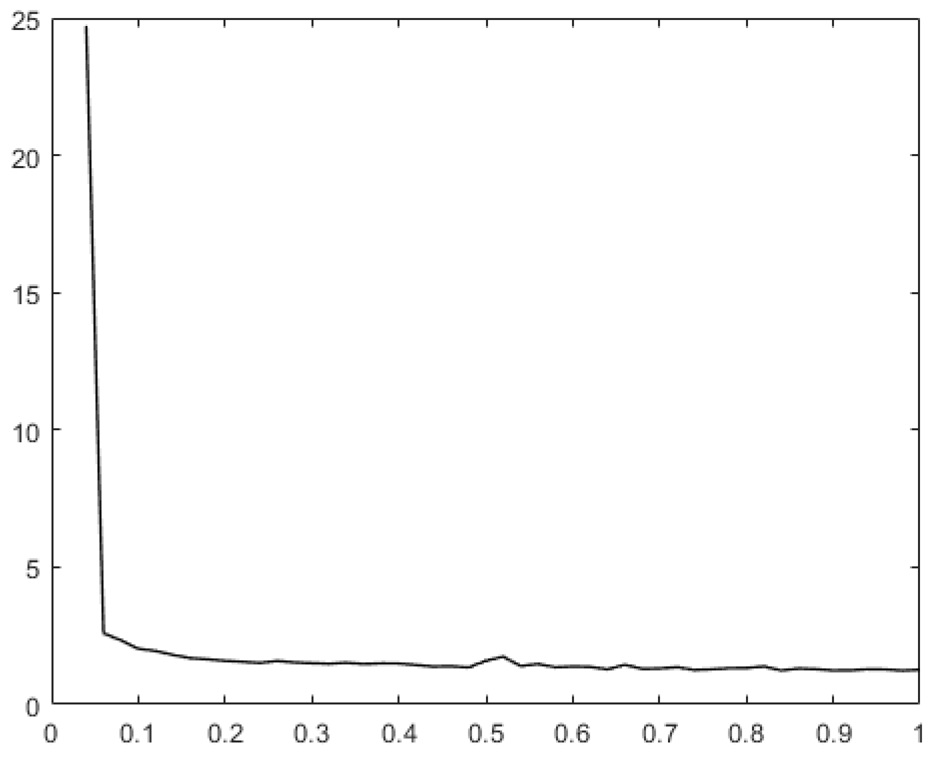
```
figure()
plot(S12.Period,S12.CSSd,'k','LineWidth',1)
drawnow;
pause(0.1)
```



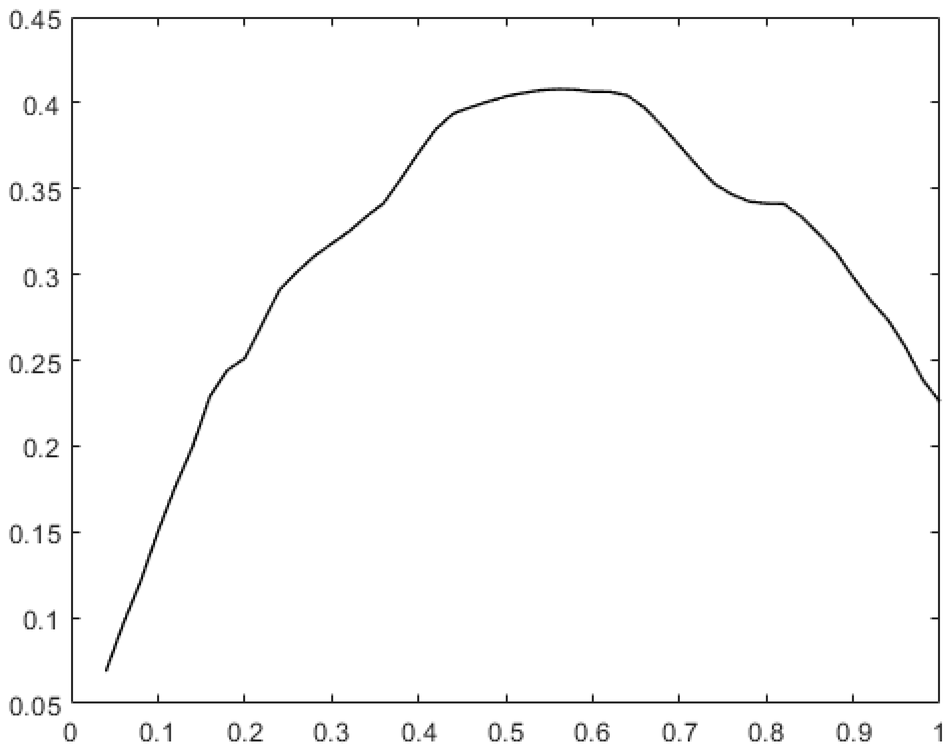
```
figure()
plot(S12.Period,S12.CSSv,'k','LineWidth',1)
drawnow;
pause(0.1)
```



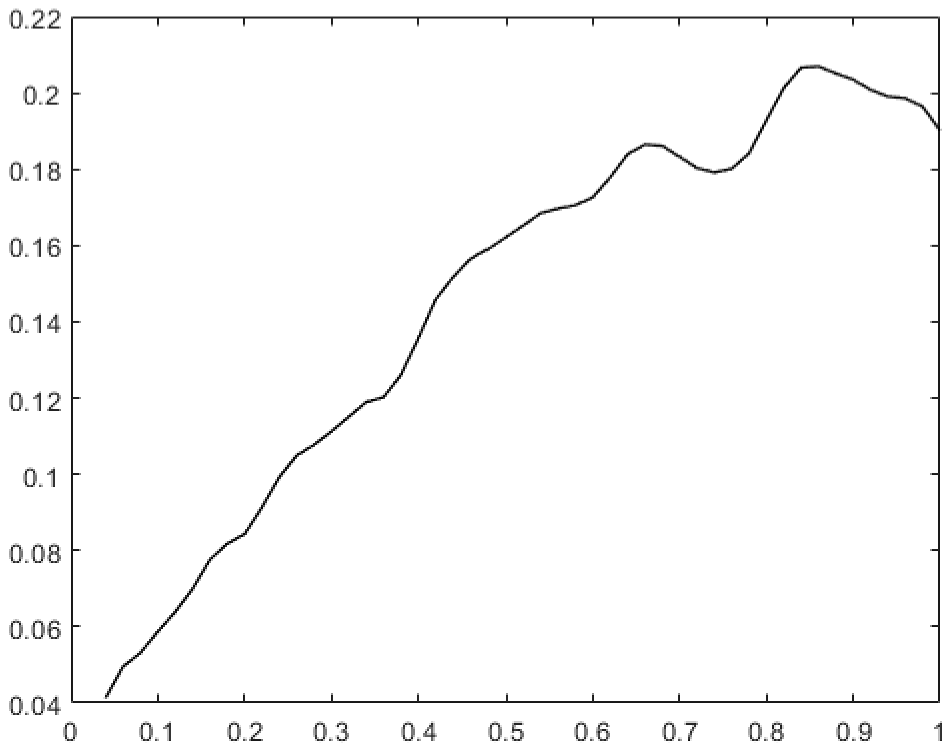
```
figure()
plot(S12.Period,S12.CSSa,'k','LineWidth',1)
drawnow;
pause(0.1)
```



```
figure()
plot(S12.Period,S12.CSSey,'k','LineWidth',1)
drawnow;
pause(0.1)
```



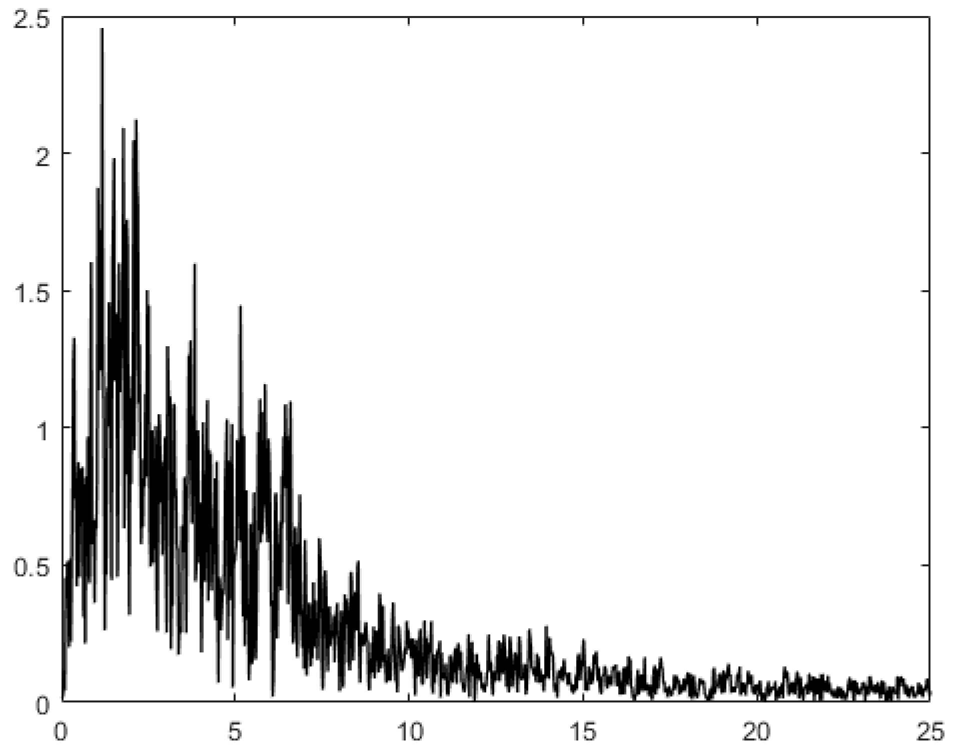
```
figure()
plot(S12.Period,S12.CSSed,'k','LineWidth',1)
drawnow;
pause(0.1)
```



Fourier amplitude spectrum and mean period

```
sw='fas';
S13=OpenSeismoMatlab(dt,xgtt,sw);
```

```
figure()
plot(S13.freq,S13.FAS,'k','LineWidth',1)
drawnow;
pause(0.1)
```



```
S13.Tm
```

```
ans =

         0.533389590787339
```

```
S13.Fm
```

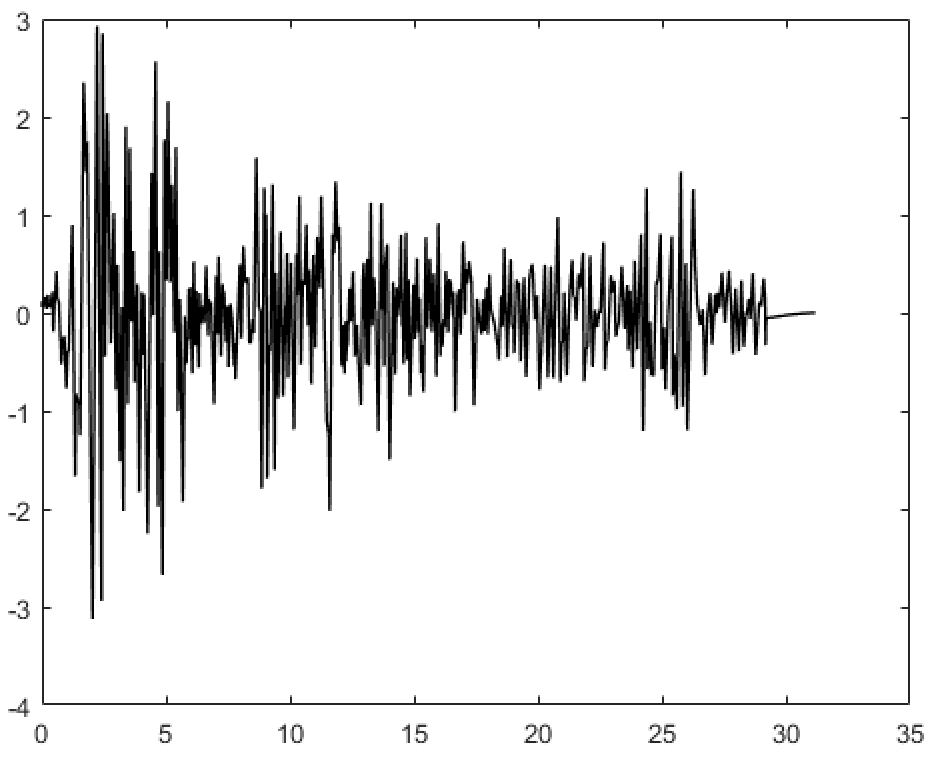
```
ans =

          3.29375516366182
```

High pass Butterworth filter

```
sw='butterworthhigh';
bOrder=4;
flc=0.1;
S14=OpenSeismoMatlab(dt,xgtt,sw,bOrder,flc);
```

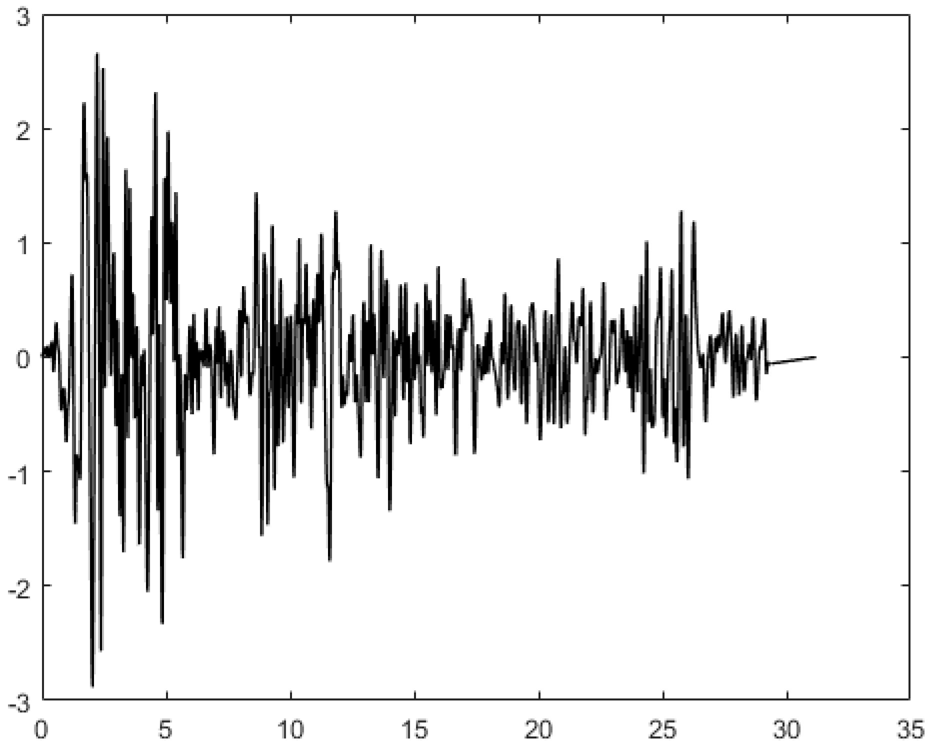
```
figure()
plot(S14.time,S14.acc,'k','LineWidth',1)
drawnow;
pause(0.1)
```



Low pass Butterworth filter

```
sw='butterworthlow';
bOrder=4;
fhc=10;
S15=OpenSeismoMatlab(dt,xgtt,sw,bOrder,fhc);
```

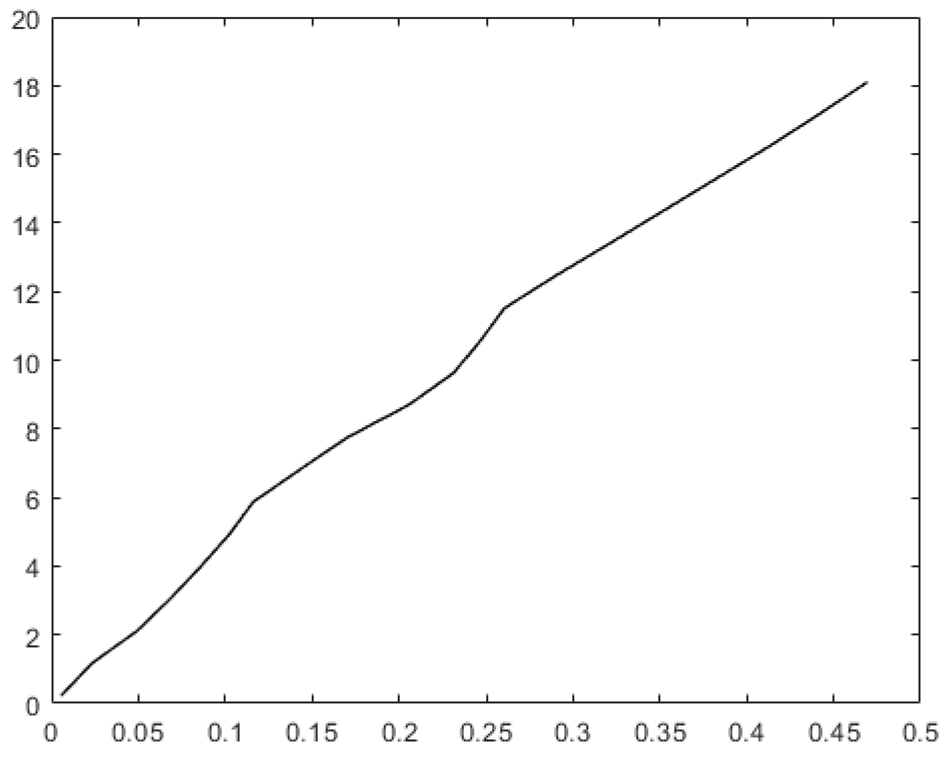
```
figure()
plot(S15.time,S15.acc,'k','LineWidth',1)
drawnow;
pause(0.1)
```



Incremental dynamic analysis

```
sw='ida';
S16=OpenSeismoMatlab(dt,xgtt,sw);
```

```
figure()
plot(S16.DM,S16.IM,'k','LineWidth',1)
drawnow;
pause(0.1)
```



Effective peak ground acceleration

```
sw='epga';
S17=OpenSeismoMatlab(dt,xgtt,sw);
```

```
S17.EPGA
```

```
ans =

          3.09546779631066
```

Spectral intensity according to Housner (1952)

```
sw='sih1952';
S18=OpenSeismoMatlab(dt,xgtt,sw);
```

```
S18.SI
```

```
ans =

          1.24373311662318
```

Spectral intensity according to Nau & Hall (1984)

```
sw='sinh1984';
S19=OpenSeismoMatlab(dt,xgtt,sw);
```

```
S19.SI
```

```
ans =

         0.530561283239986
```

Copyright

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

Published with MATLAB® R2022b

example LEReSp

Calculate linear elastic response spectra in OpenSeismoMatlab

Contents

Generate earthquake motion

For reproducibility

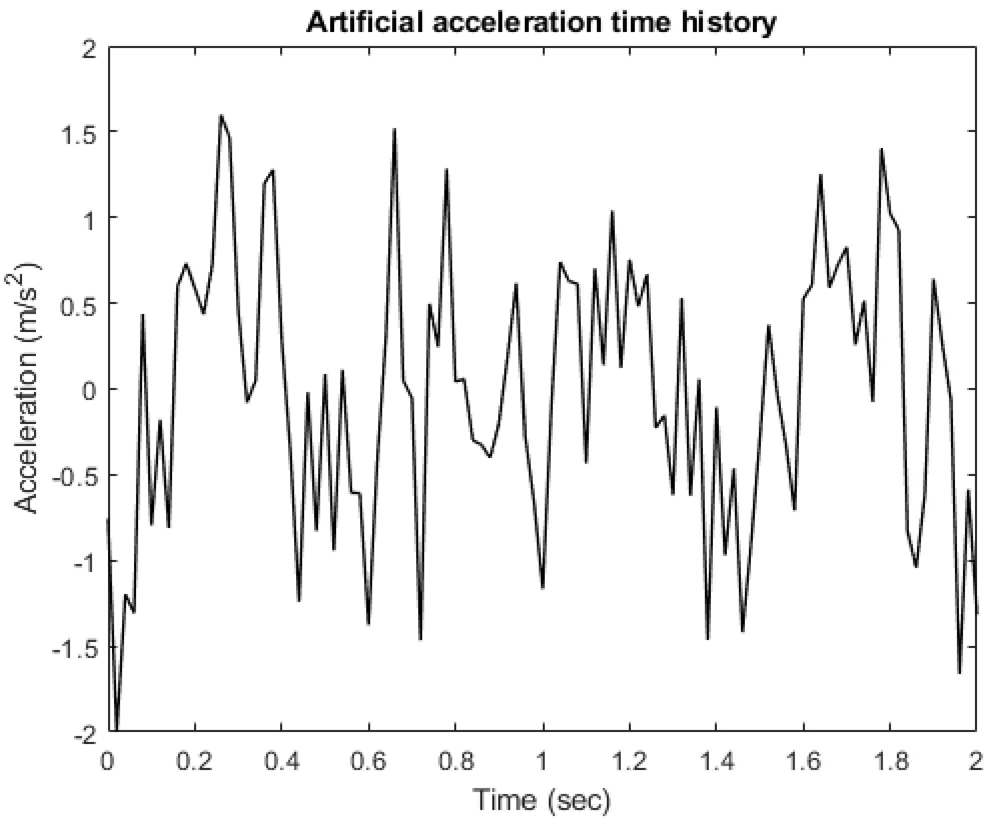
```
rng(0)
```

Generate earthquake acceleration time history

```
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
```

Plot the generated time history

```
figure()
plot(t,xgtt,'k','LineWidth',1)
ylabel('Acceleration (m/s^2)')
xlabel('Time (sec)')
title('Artificial acceleration time history')
drawnow;
pause(0.1)
```



Setup parameters for LEReSp function

Eigenperiods

```
T=logspace(log10(0.02),log10(5),1000)';
```

Critical damping ratio

```
ksi=0.05;
```

Maximum ratio of the integration time step to the eigenperiod

```
dtTol=0.02;
```

Algorithm to be used for the time integration

```
AlgID='U0-V0-Opt';
```

Minimum absolute value of the eigenvalues of the amplification matrix

```
rinf=1;
```

Calculate spectra and pseudospectra

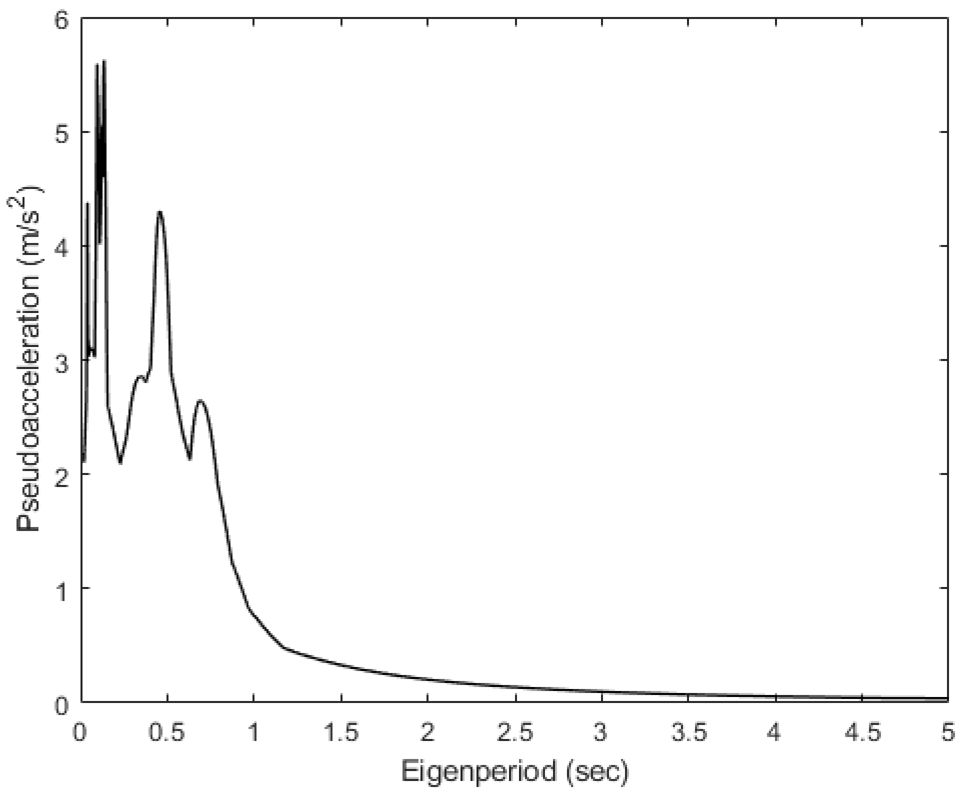
Apply LEReSp

```
[PSa,PSv,Sd,Sv,Sa,Siev]=LEReSp(dt,xgtt,T,ksi,dtTol,AlgID,rinf);
```

Plot the spectra and pseudospectra

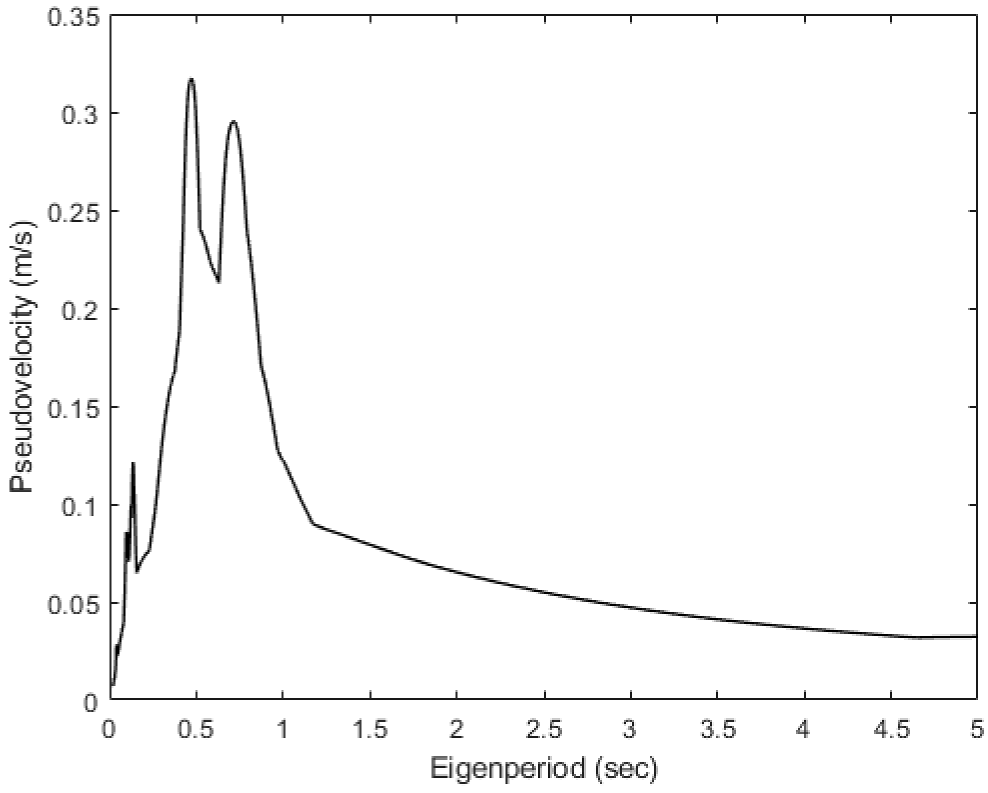
Pseudoacceleration spectrum

```
figure()
plot(T,PSa,'k','LineWidth',1)
ylabel('Pseudoacceleration (m/s^2)')
xlabel('Eigenperiod (sec)')
drawnow;
pause(0.1)
```



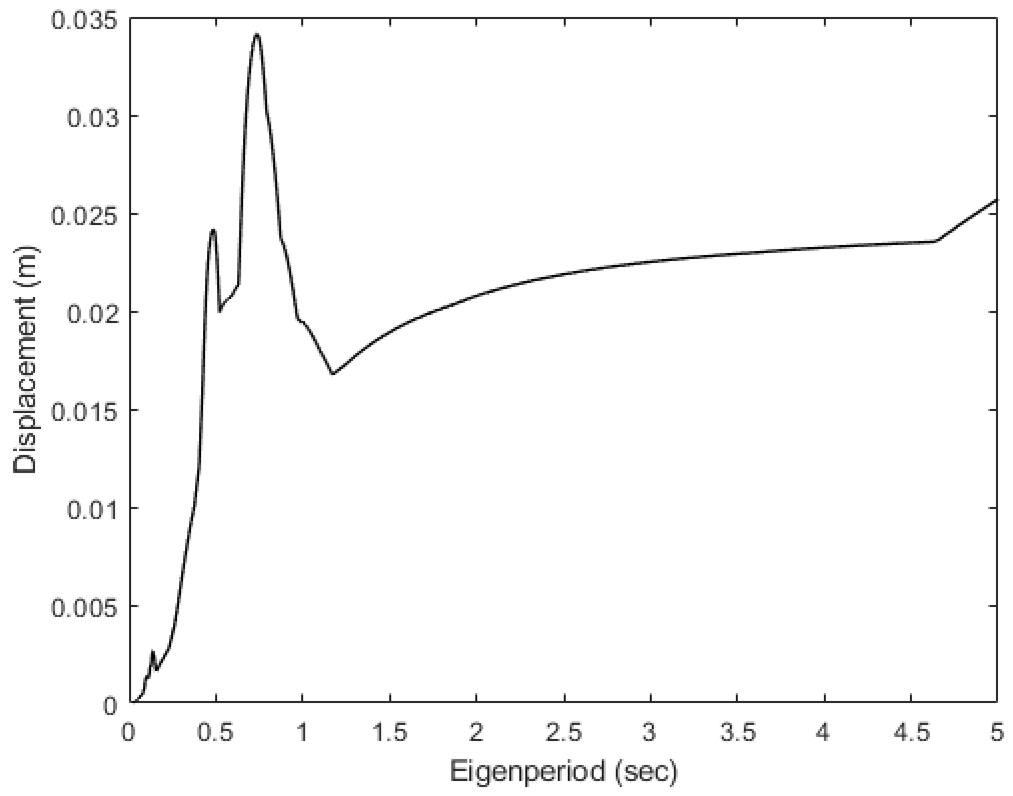
Pseudovelocity spectrum

```
figure()
plot(T,PSv,'k','LineWidth',1)
ylabel('Pseudovelocity (m/s)')
xlabel('Eigenperiod (sec)')
drawnow;
pause(0.1)
```



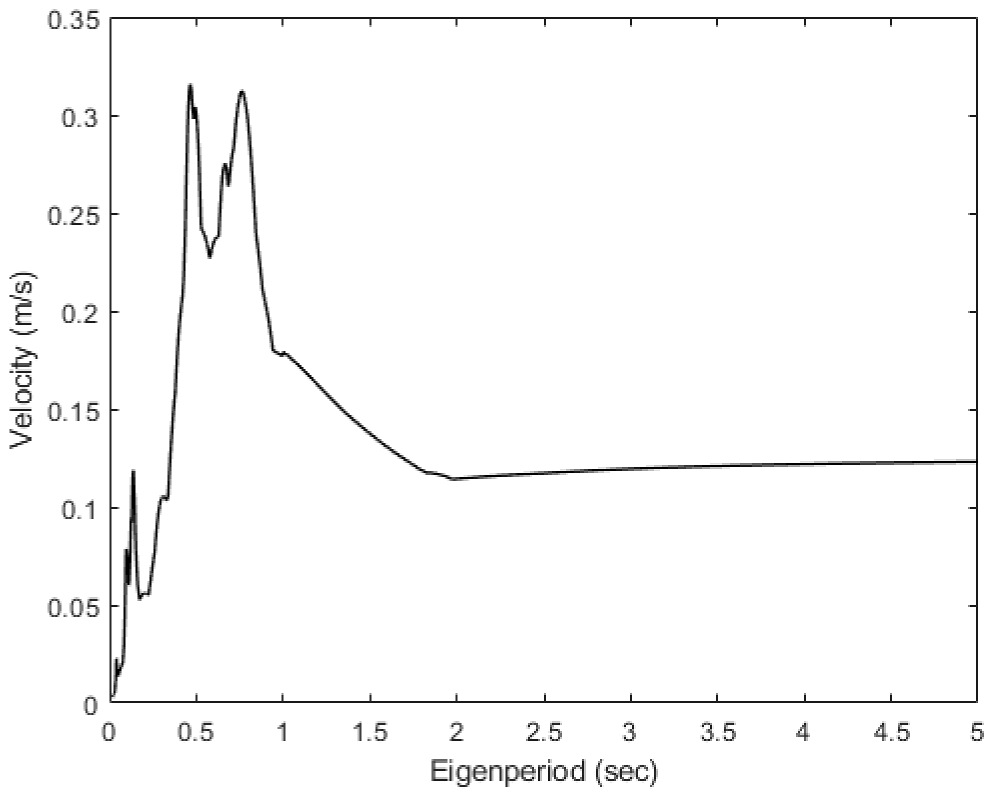
Displacement spectrum

```
figure()
plot(T,Sd,'k','LineWidth',1)
ylabel('Displacement (m)')
xlabel('Eigenperiod (sec)')
drawnow;
pause(0.1)
```



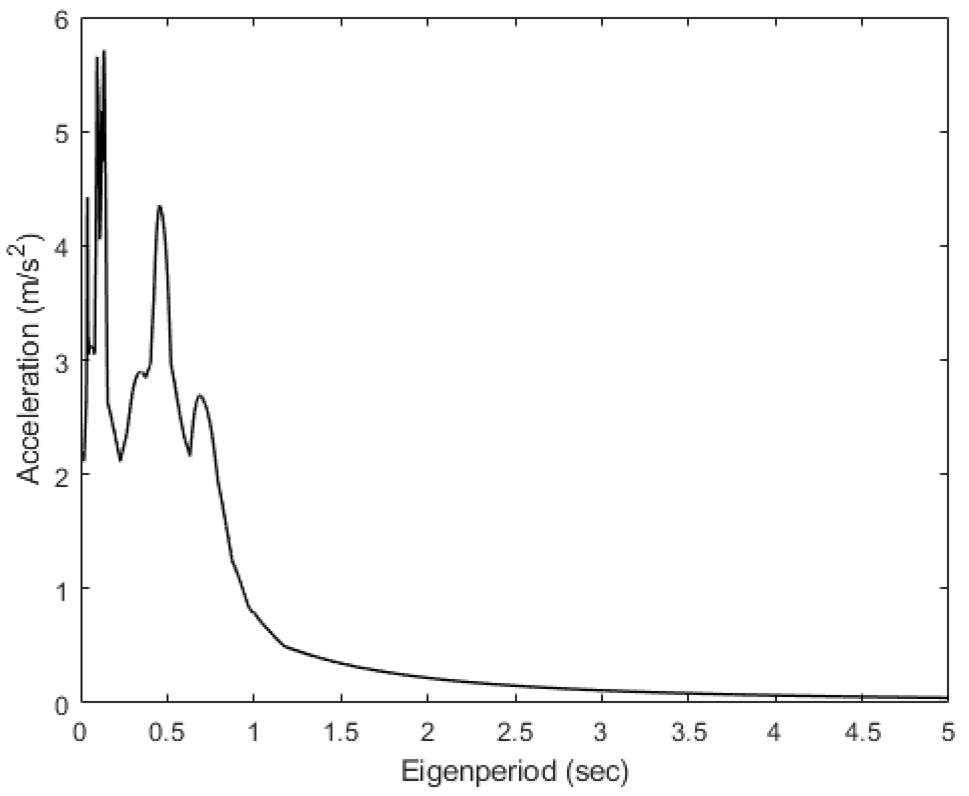
Velocity spectrum

```
figure()
plot(T,Sv,'k','LineWidth',1)
ylabel('Velocity (m/s)')
xlabel('Eigenperiod (sec)')
drawnow;
pause(0.1)
```



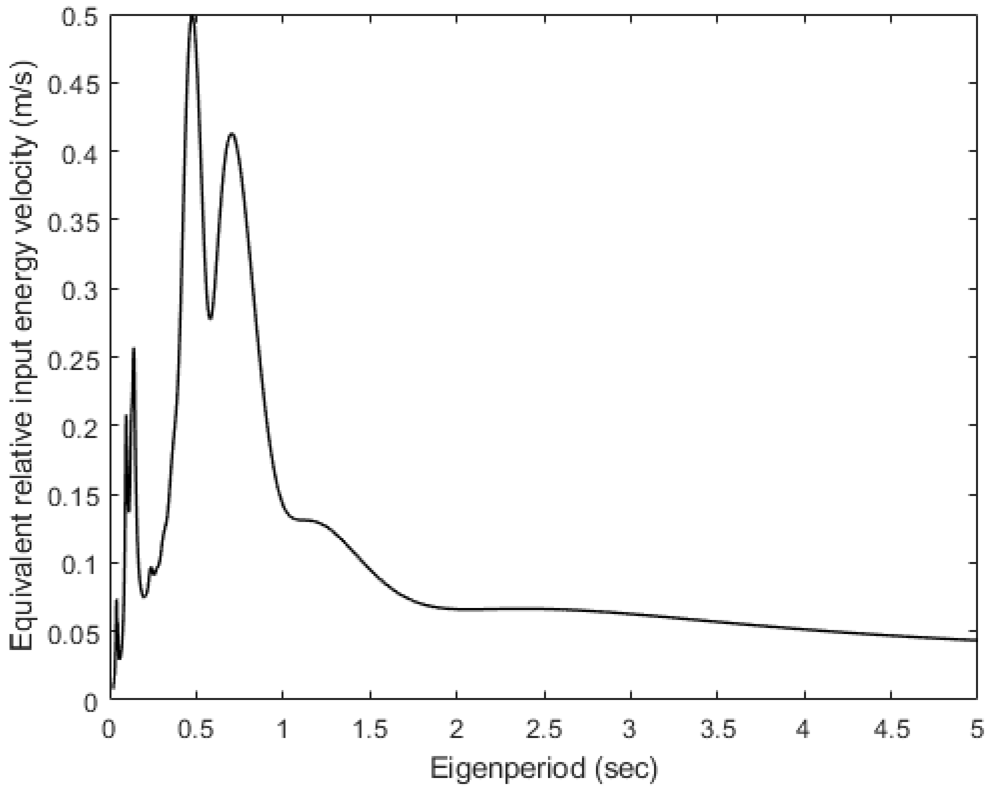
Acceleration spectrum

```
figure()
plot(T,Sa,'k','LineWidth',1)
ylabel('Acceleration (m/s^2)')
xlabel('Eigenperiod (sec)')
drawnow;
pause(0.1)
```



Equivalent relative input energy velocity

```
figure()
plot(T,Siev,'k','LineWidth',1)
ylabel('Equivalent relative input energy velocity (m/s)')
xlabel('Eigenperiod (sec)')
drawnow;
pause(0.1)
```



Copyright

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

Published with MATLAB® R2022b

example 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';
    'elcentro_NS_trunc'; % 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;
```

Extract linear elastic response spectra

```
sw='elrs';
```

Calculation

Initialize LERS

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

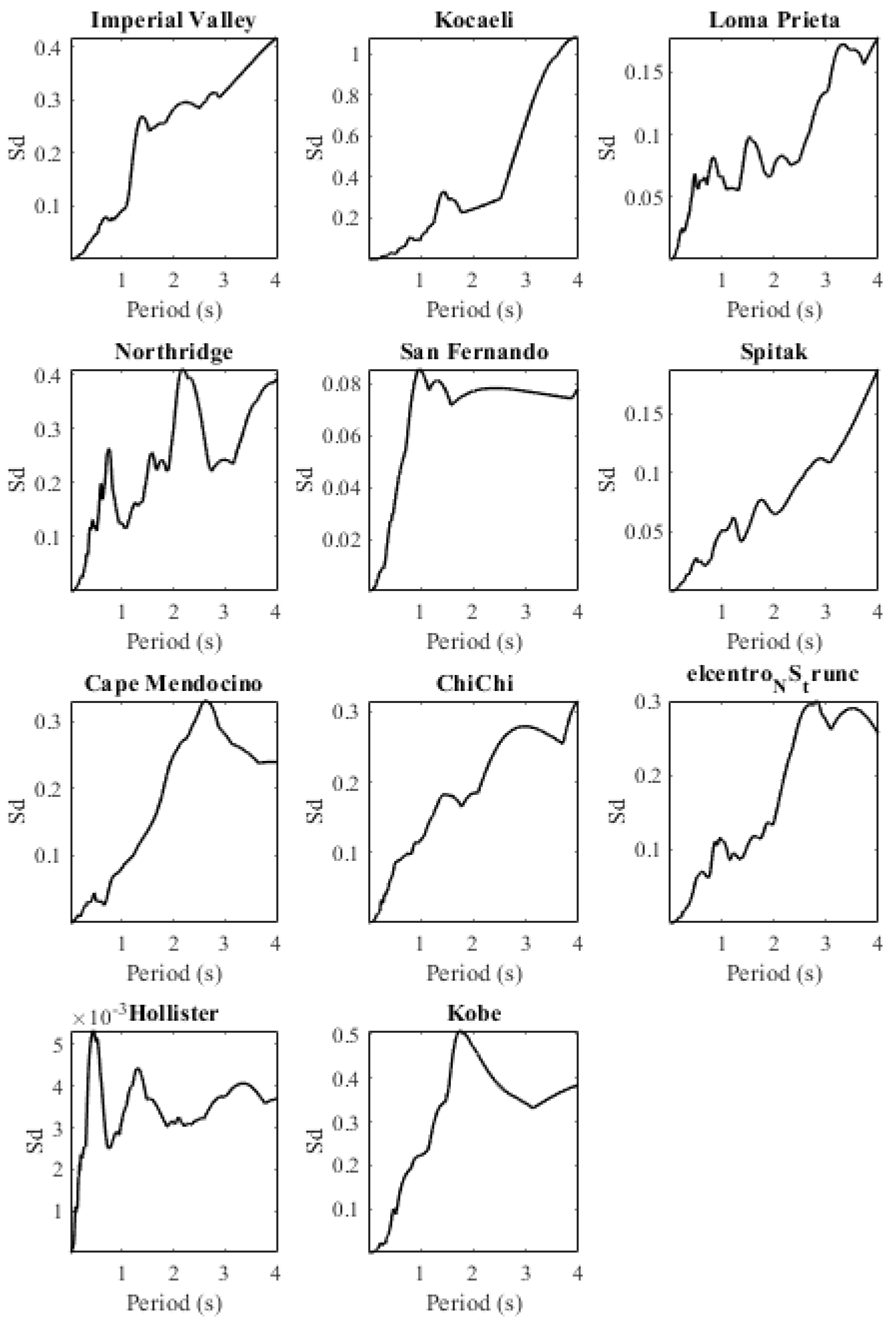
Calculation of peak values

```
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,Tspectra,ksi);
    LERS{i}=[S.Period,S.Sd,S.PSv,S.PSa];
end
```

Output

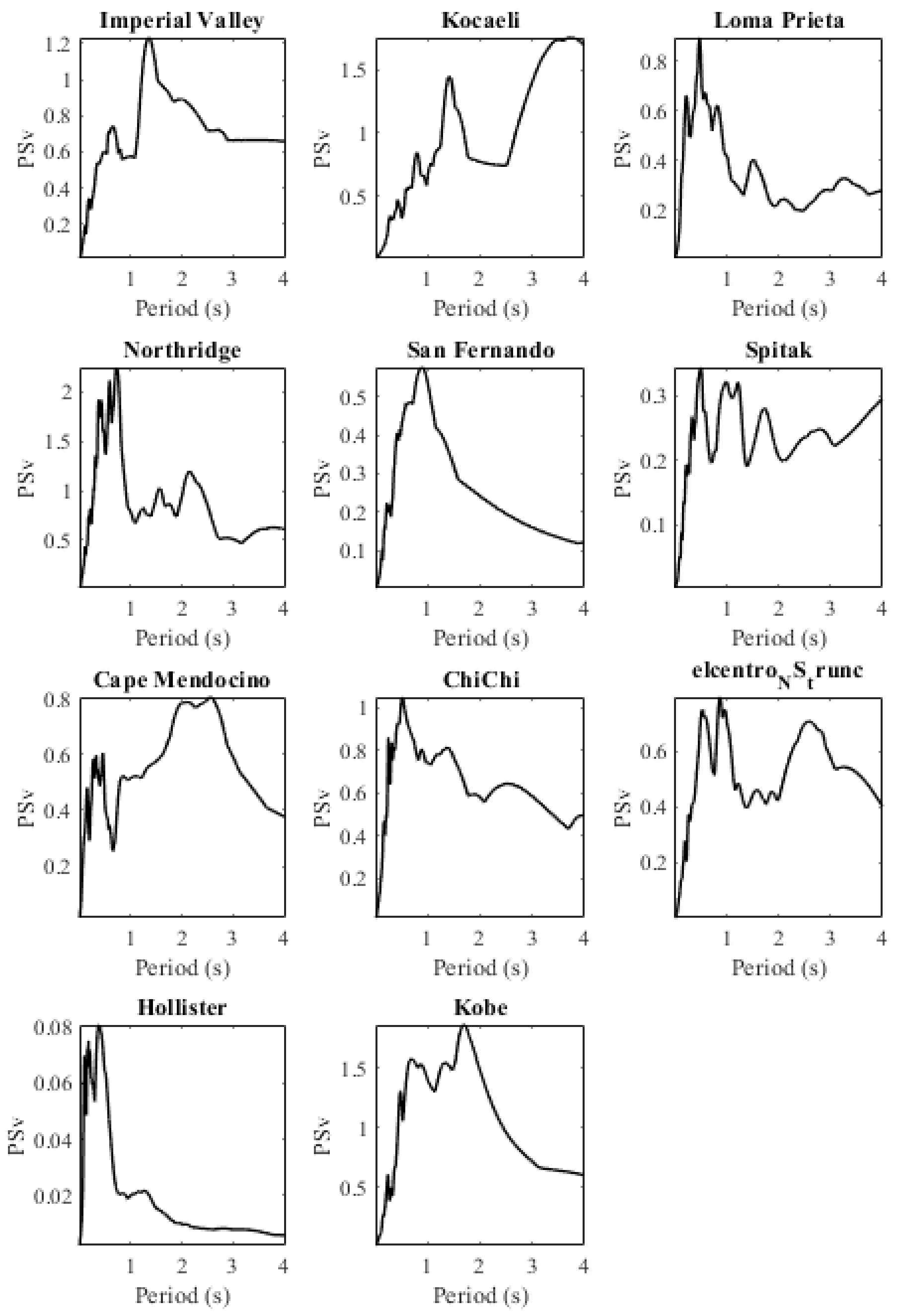
Plot displacement response spectra

```
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
drawnow;
pause(0.1)
```



Plot pseudo-velocity response spectra

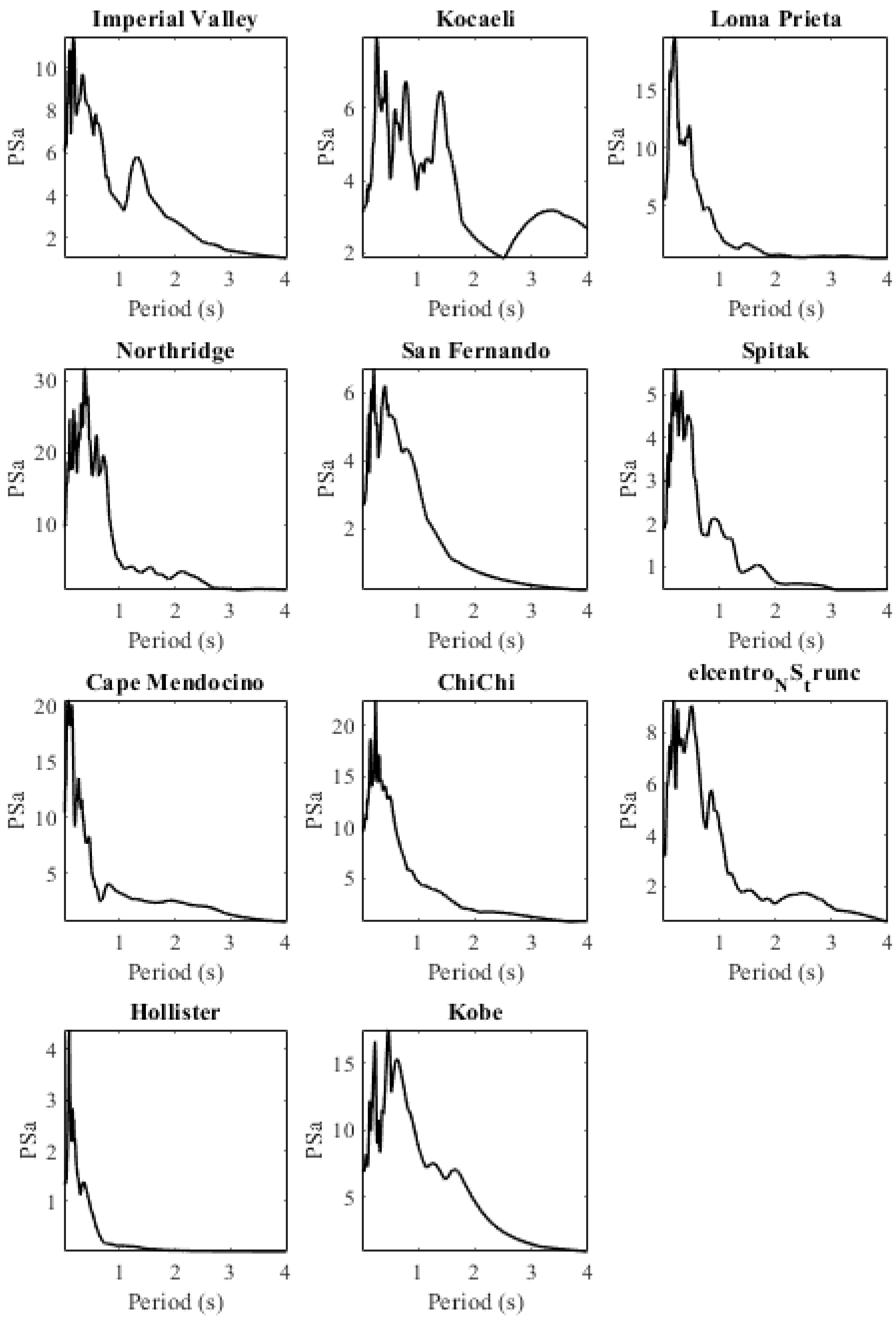
```
Fig2 = figure('units', 'centimeters', 'Position', [0,0, 20/sqrt(2), 20]);
% Scan all subplots
for i=1:numel(eqmotions)
    subplot(4,3,i)
    plot(LERS{i}(:,1),LERS{i}(:,3),'k','LineWidth',1);
    set(gca,'FontName','Times New Roman')
    title(eqmotions{i},'FontName','Times New Roman')
    ylabel('PSv','FontName','Times New Roman')
    xlabel('Period (s)','FontName','Times New Roman')
    axis tight
end
drawnow;
pause(0.1)
```



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
drawnow;
pause(0.1)
```



Copyright

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

Published with MATLAB® R2022b

example PulseDecomp

Pulse decomposition of a velocity time history in OpenSeismoMatlab

Contents

Generate earthquake motion

For reproducibility

```
rng(0)
```

Generate earthquake velocity time history with a pulse

```
duration = 10;
dt = 0.01;
t = linspace(0, duration, duration /dt)';
num_sin_components = 3;
xgt = zeros(length(t),1);
for i = 1:num_sin_components
    s = rand * sin(2 * pi * 0.5*rand * t + rand);
    xgt = xgt + s;
end
```

Setup parameters for PulseDecomp function

Wavelet family to be considered as the contained pulse

```
wname = 'db4';
```

Minimum pulse period for the continuous 1-D wavelet transform

```
TpMin = 0.25;
```

Minimum pulse period for the continuous 1-D wavelet transform

```
TpMax = 15;
```

Number of pulse period values between TpMin and TpMax

```
nScales = 50;
```

Calculate Pulse and residual motion

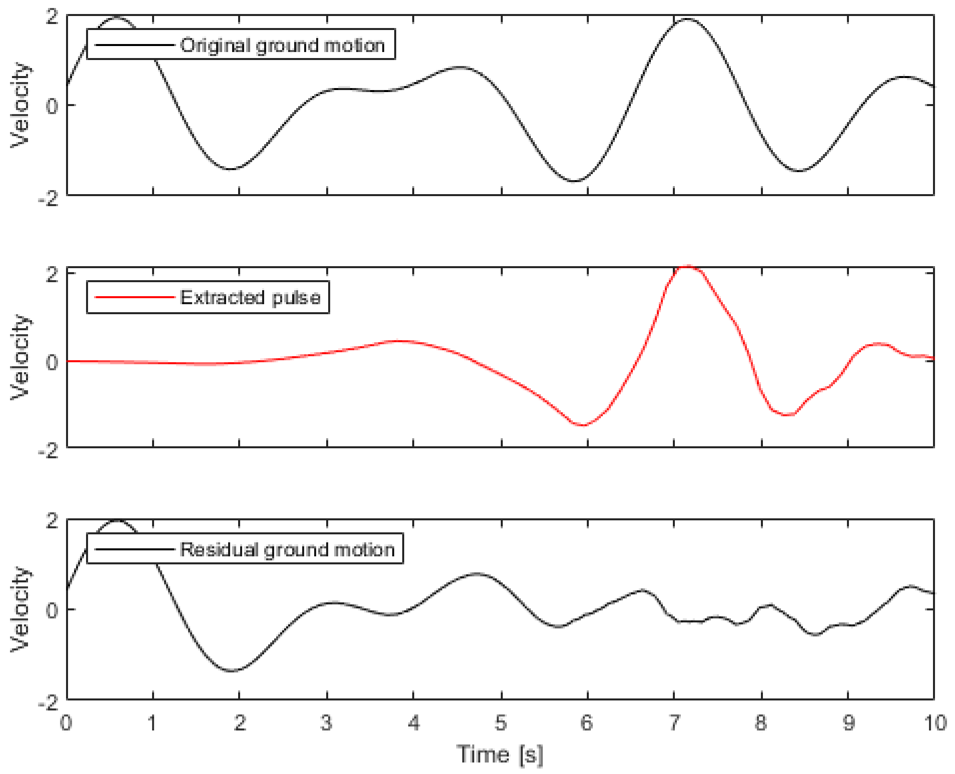
Apply PulseDecomp

```
[pulseTH,resTH,Tp,wavScale,wavCoef] = ...
    PulseDecomp(dt,xgt,wname,TpMin,TpMax,nScales);
```

Plot the initial, pulse and residual time histories

Initialize figure

```
np = length(xgt);
time = dt:dt:dt*np;
fig = figure();
% Initial velocity time history
subplot(3,1,1)
plot(time, xgt, '-k')
legend('Original ground motion','location','northwest')
ylabel('Velocity');
set(gca, 'xticklabel', [])
% Pulse velocity time history
subplot(3,1,2)
plot(time, pulseTH, '-r')
legend('Extracted pulse','location','northwest')
ylabel('Velocity');
set(gca, 'xticklabel', [])
% Residual velocity time history
subplot(3,1,3)
plot(time, resTH , '-k')
legend('Residual ground motion','location','northwest')
hy = ylabel('Velocity');
hx = xlabel('Time [s]');
drawnow;
pause(0.1)
```



Copyright

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

Published with MATLAB® R2022b

example RPSReSp

Calculate rigid plastic sliding response spectra in OpenSeismoMatlab

Contents

Generate earthquake motion

For reproducibility

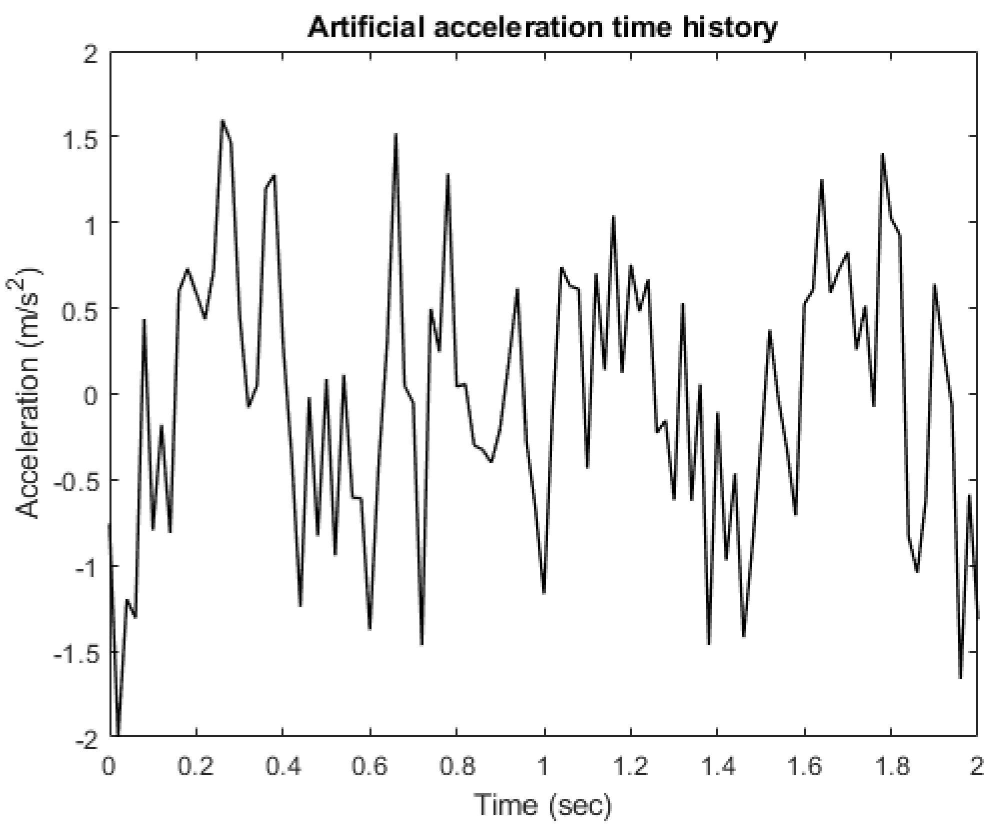
```
rng(0)
```

Generate earthquake acceleration time history

```
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
```

Plot the generated time history

```
figure()
plot(t,xgtt,'k','LineWidth',1)
ylabel('Acceleration (m/s^2)')
xlabel('Time (sec)')
title('Artificial acceleration time history')
drawnow;
pause(0.1)
```



Setup parameters for RPSReSp function

Coulomb friction coefficients

```
CF=linspace(0.05,0.5,1000)';
```

Algorithm to be used for the time integration

```
AlgID='U0-V0-Opt';
```

Minimum absolute value of the eigenvalues of the amplification matrix

```
rinf=1;
```

Maximum tolerance of convergence for time integration algorithm

```
maxtol=0.01;
```

Maximum number of iterations per integration time step

```
jmax=200;
```

Infinitesimal acceleration

```
dak=eps;
```

Calculate spectra

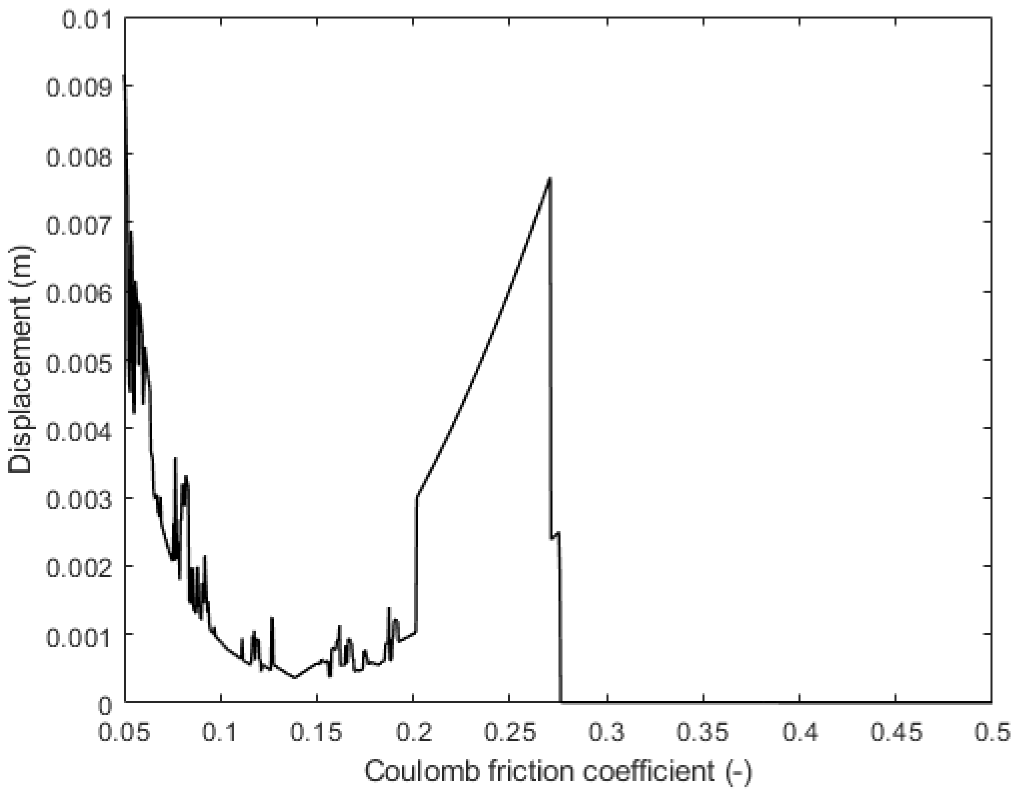
Apply RPSReSp

```
[Sd,Sv,Sa]=RPSReSp(dt,xgtt,CF,AlgID,rinf,maxtol,jmax,dak);
```

Plot the spectra

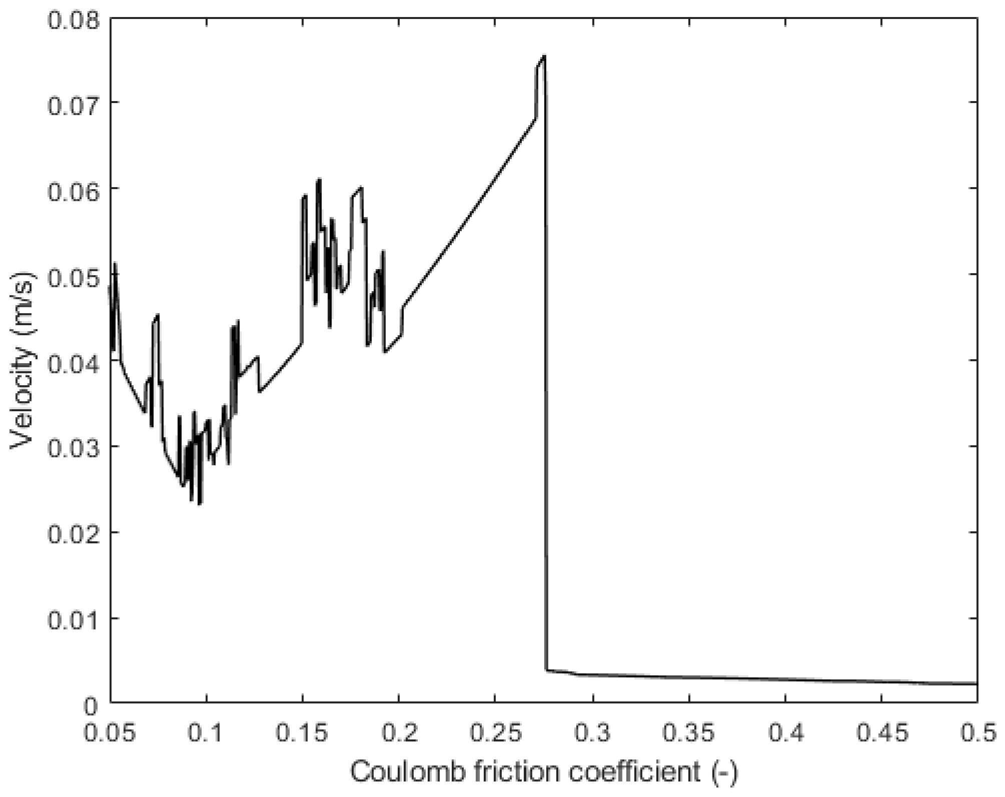
Displacement spectrum

```
figure()
plot(CF,Sd,'k','LineWidth',1)
ylabel('Displacement (m)')
xlabel('Coulomb friction coefficient (-)')
drawnow;
pause(0.1)
```



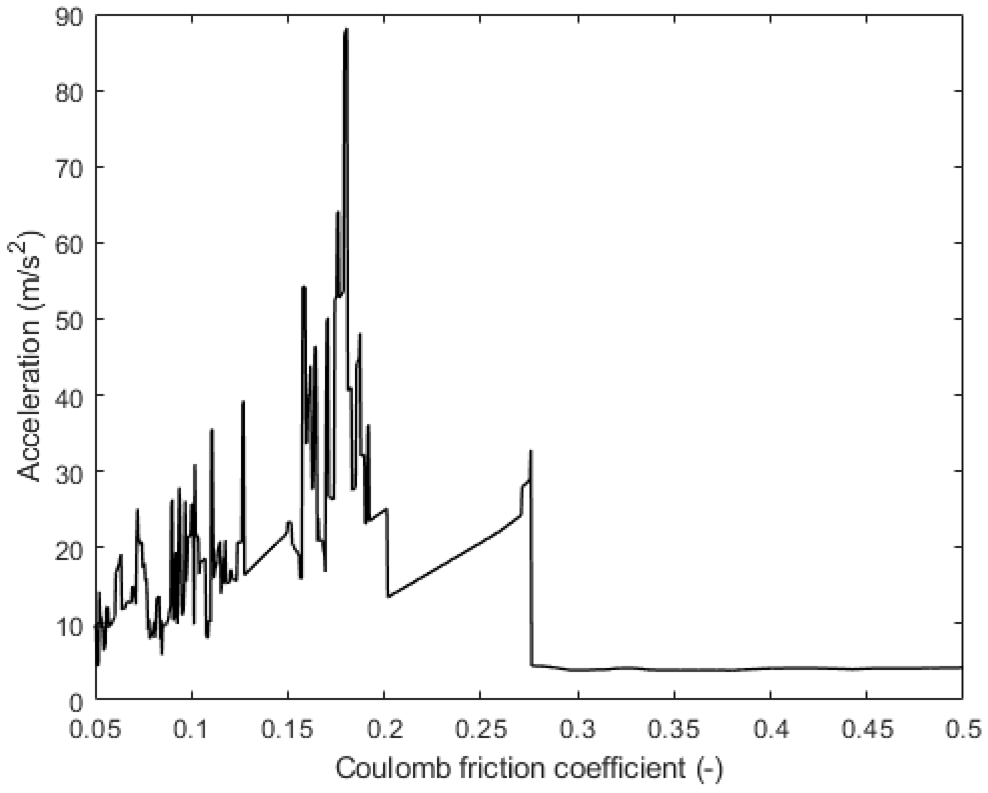
Velocity spectrum

```
figure()
plot(CF,Sv,'k','LineWidth',1)
ylabel('Velocity (m/s)')
xlabel('Coulomb friction coefficient (-)')
drawnow;
pause(0.1)
```



Acceleration spectrum

```
figure()
plot(CF,Sa,'k','LineWidth',1)
ylabel('Acceleration (m/s^2)')
xlabel('Coulomb friction coefficient (-)')
drawnow;
pause(0.1)
```



Copyright

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

Published with MATLAB® R2022b

example Comparison of elastic and constant ductility response spectra for mu=1

Compare the linear elastic acceleration response spectrum and the constant ductility response spectrum for mu=1 (degenerates to linear elastic). Also, compare with corresponding results of SeismoSignal software

Contents

Input

Earthquake motion

```
eqmotions={'Imperial Valley'}; % Imperial valley 1979
data=load([eqmotions{1},'.dat']);
t=data(:,1);
dt=t(2)-t(1);
xgtt=data(:,2);
```

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

```
Tspectra=(0.05:0.05:4)';
```

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

```
ksi=0.05;
```

First we use the CDS to calculate linear response

Set the target ductility

```
mu=1; % mu=1 equivalent to a linear SDoF
```

Extract constant ductility response spectra

```
sw='cdrs';
```

Calculation CDRS{i}=[S.Period,S.CDSd,S.CDSv,S.CDPSa,S.fyK,S.muK,S.iterK];

```
S1=OpenSeismoMatlab(dt,xgtt,sw,Tspectra,ksi,mu);
```

Next we use ES to calculate linear response

Extract linear elastic response spectra

```
sw='elrs';
```

Initialize LERS

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

Calculation LERS{i}=[S.Period,S.Sd,S.PSv,S.PSa];

```
S2=OpenSeismoMatlab(dt,xgtt,sw,Tspectra,ksi);
```

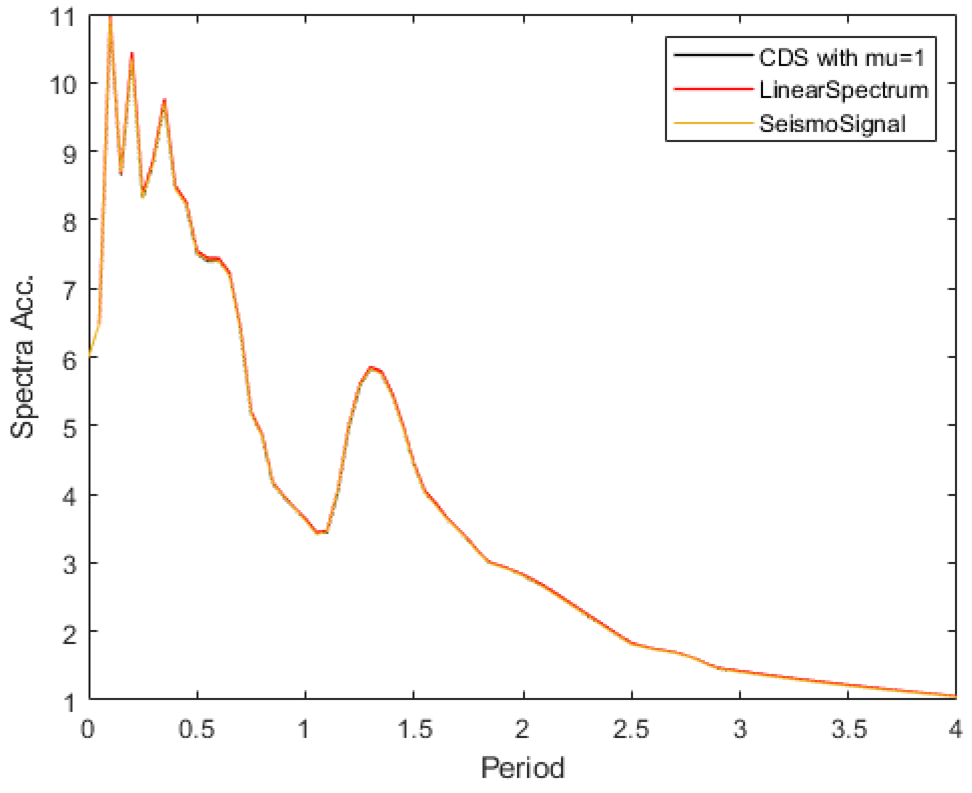
Results from SeismoSignal

```
fileID = fopen('SeismoSignal_Imperial Valley.txt');
for idx = 1:5
    fgetl(fileID);
end
C = textscan(fileID,repmat('%f',1,12));
fclose(fileID);
```

Output

Plot spectral acceleration response spectra

```
plot(S1.Period,S1.CDSa,'k','LineWidth',1);
hold on
plot(S2.Period,S2.Sa,'r','LineWidth',1);
plot(C{1},C{2})
legend('CDS with mu=1','LinearSpectrum','SeismoSignal');
xlabel('Period');
ylabel('Spectra Acc.')
```



Copyright

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

Published with MATLAB® R2022b

example Comparison of constant ductility response spectra of RSN1044 for mu=2

Contents

Input

Earthquake motions

```
GMFileLoc = 'input\';
```

A text file containing the lists of GM names

```
RecordList = ['input\Record_List.txt'];
```

Read E.Q. records name from the specifid list

```
fid = fopen (RecordList,'r');
records = textscan(fid, '%s'); % length = num of records*1, cell
fclose (fid);
```

Read E.Q. records name from the specifid list Return variables include record names, dt, num of points, PGA, Acc TH, Spectral Acc The accelration-related variables (PGA, Sa, Acc TH) are in unit of g; disp-related in m If you do not specify scaling factors at this step, the default value = 1 for each GM

```
[RecordName_all, dt, numPoint_all, PGA_all, xgtt] =...
    fn_ParseAT2File2(GMFileLoc, records);
```

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

```
Tspectra=(0.05:0.05:4)';
```

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

```
ksi=0.02;
```

Results from OpenSeismoMatlab

Set the target ductility

```
mu=2; % mu=1 equivalent to a linear SDoF
```

Extract constant ductility response spectra

```
sw='cdrs';
```

Calculation CDRS{i}=[S.Period,S.CDSd,S.CDSv,S.CDPSa,S.fyK,S.muK,S.iterK];

```
S1=OpenSeismoMatlab(dt,xgtt{1},sw,Tspectra,ksi,mu);
```

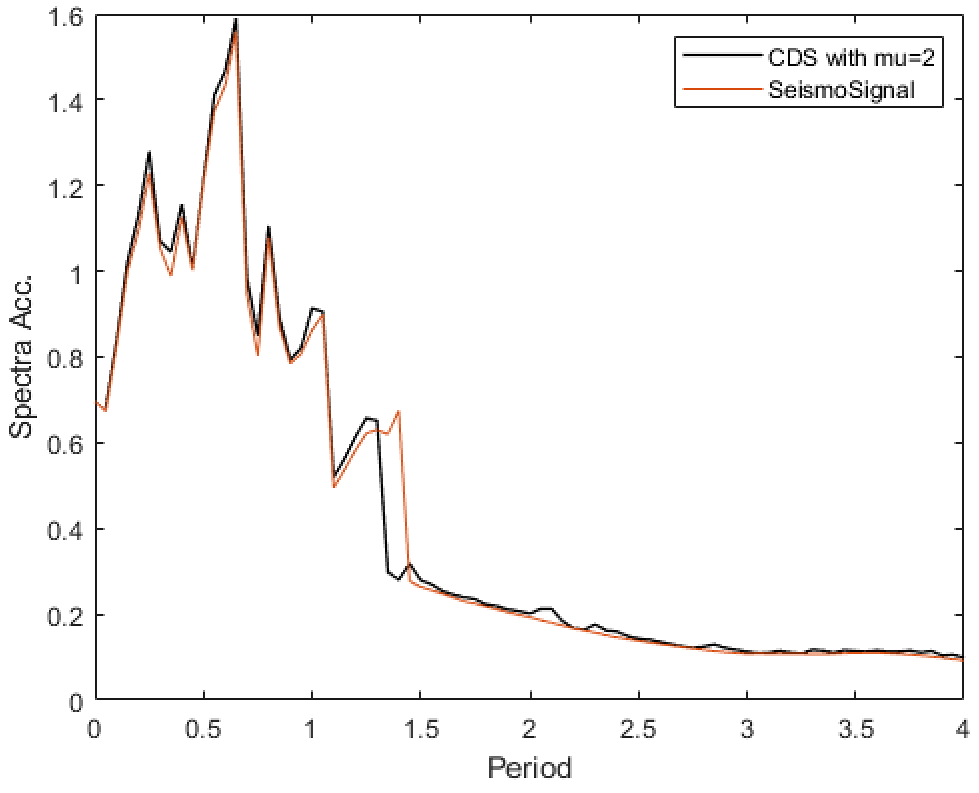
Results from SeismoSignal

```
fileID = fopen('input/RSN1044_Damping2.txt');
for idx = 1:5
    fgetl(fileID);
end
C = textscan(fileID,repmat('%f',1,15));
fclose(fileID);
```

Output

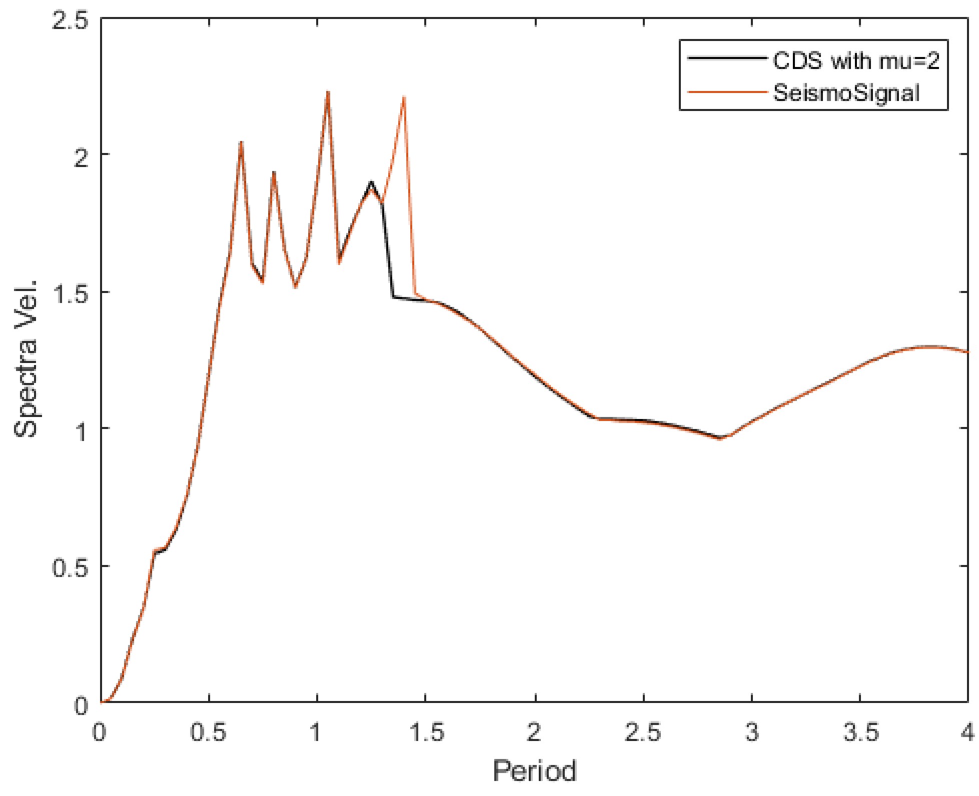
Plot spectral acceleration response spectra

```
figure()
plot(S1.Period,S1.CDSa/9.8,'k','LineWidth',1);
hold on
%plot(S2.Period,S2.Sa,'r','LineWidth',1);
plot(C{1},C{3})
legend('CDS with mu=2','SeismoSignal');
xlabel('Period');
ylabel('Spectra Acc.')
hold off
drawnow;
pause(0.1)
```



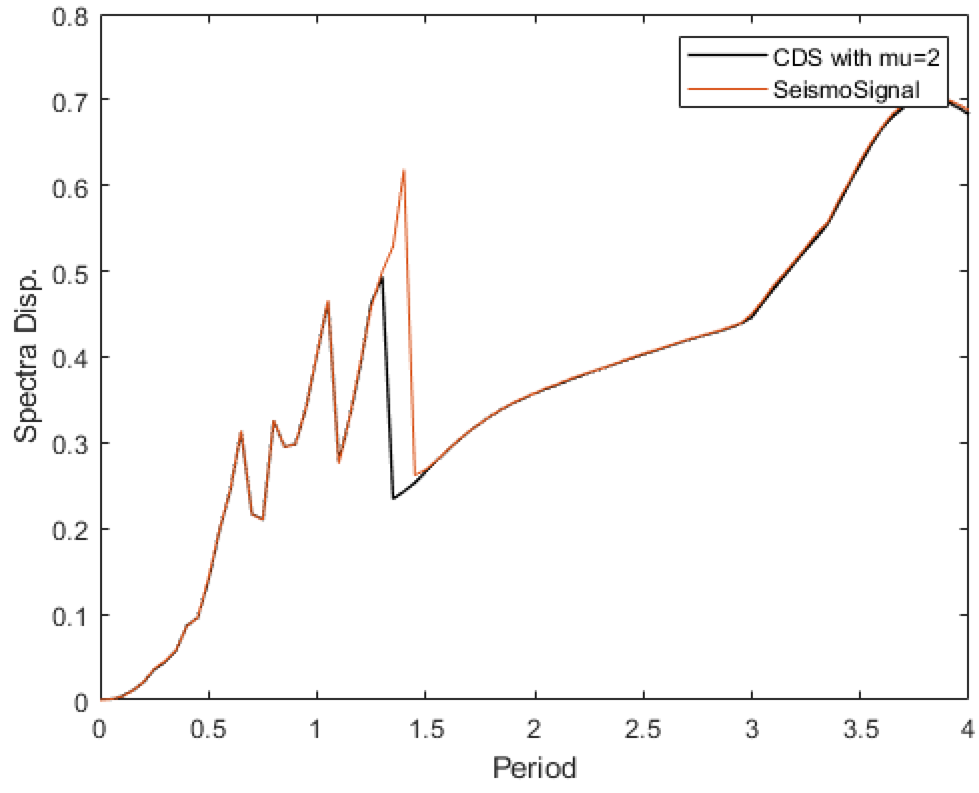
Plot spectral velocity response spectra

```
figure()
plot(S1.Period,S1.CDSv,'k','LineWidth',1);
hold on
%plot(S2.Period,S2.Sa,'r','LineWidth',1);
plot(C{1},C{7}/100)
legend('CDS with mu=2','SeismoSignal');
xlabel('Period');
ylabel('Spectra Vel.')
hold off
drawnow;
pause(0.1)
```



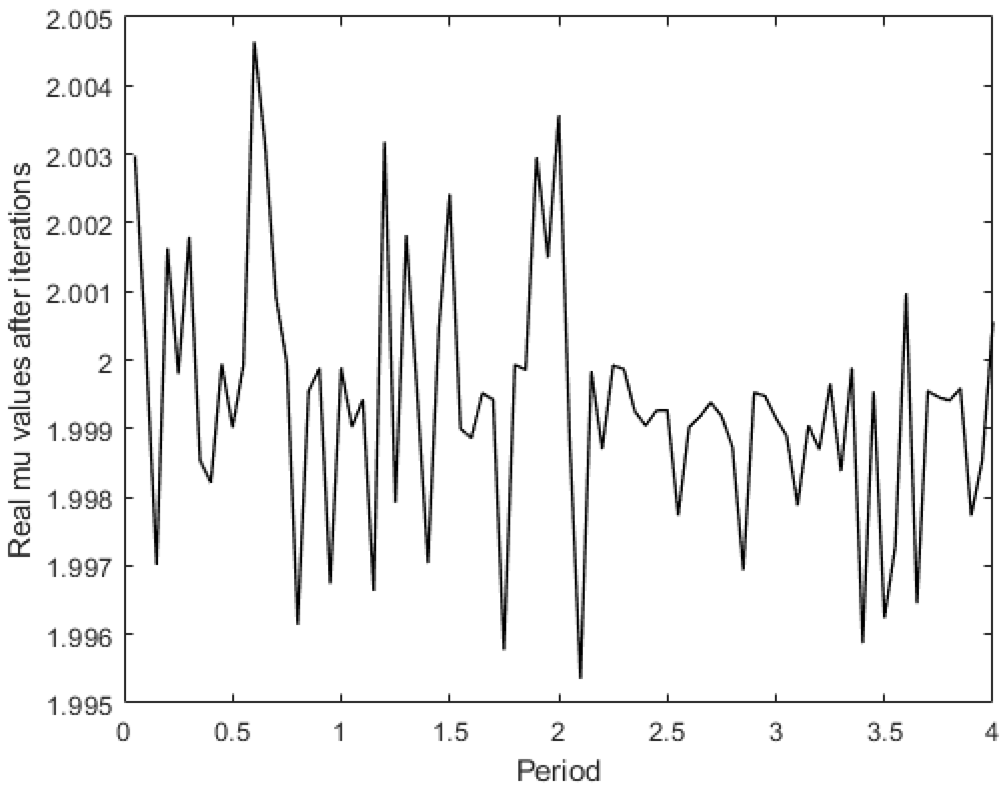
Plot spectral displacement response spectra

```
figure()
plot(S1.Period,S1.CDSd,'k','LineWidth',1);
hold on
%plot(S2.Period,S2.Sa,'r','LineWidth',1);
plot(C{1},C{11}/100)
legend('CDS with mu=2','SeismoSignal');
xlabel('Period');
ylabel('Spectra Disp.')
hold off
drawnow;
pause(0.1)
```



Plot the achieved ductility for each eigenperiod. It should be equal or close to the target ductility mu=2

```
figure()
plot(S1.Period,S1.muK,'k','LineWidth',1);
xlabel('Period');
ylabel('Real mu values after iterations')
drawnow;
pause(0.1)
```



Copyright

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

Published with MATLAB® R2022b

verification Constant ductility response spectrum

Contents

Reference

Chopra, A. K. (2020). Dynamics of structures, Theory and Applications to Earthquake Engineering, 5th edition. Prenctice Hall.

Description

The constant-ductility response spectra of Figure 7.5.2 of the above reference are verified. Elastic-perfectly plastic systems are considered. Damping is equal to 5% of its critical value. Target ductility factors equal to 1,1.5,2,4 and 8 are considered.

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

Calculate constant ductility response spectra of earthquake motion

Switch

```
sw='cdrs';
```

Eigenperiods

```
T=logspace(log10(0.05),log10(3),20);
```

Critical damping ratio

```
ksi=0.05;
```

Ductilities

```
mu1=1;
```

```
mu2=1.5;
mu3=2;
mu4=4;
mu5=8;
```

Post-yield stiffness factor

```
pysf=0.001;
```

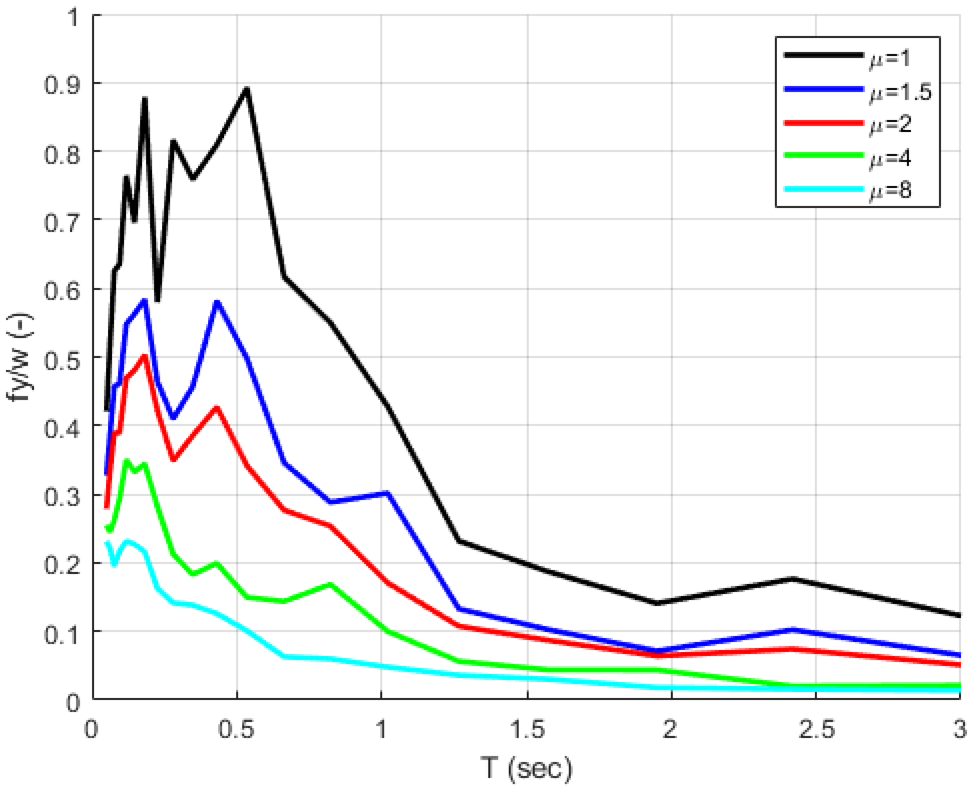
Apply OpenSeismoMatlab once for each target ductility value

```
S1=OpenSeismoMatlab(dt,xgtt,sw,T,ksi,mu1,pysf);
S2=OpenSeismoMatlab(dt,xgtt,sw,T,ksi,mu2,pysf);
S3=OpenSeismoMatlab(dt,xgtt,sw,T,ksi,mu3,pysf);
S4=OpenSeismoMatlab(dt,xgtt,sw,T,ksi,mu4,pysf);
S5=OpenSeismoMatlab(dt,xgtt,sw,T,ksi,mu5,pysf);
```

Plot the constant ductility response spectra

Initialize figure

```
figure()
hold on
% Plot the constant ductility response spectra
plot(S1.Period,S1.fyK/9.81, 'k-', 'LineWidth', 2)
plot(S2.Period,S2.fyK/9.81, 'b-', 'LineWidth', 2)
plot(S3.Period,S3.fyK/9.81, 'r-', 'LineWidth', 2)
plot(S4.Period,S4.fyK/9.81, 'g-', 'LineWidth', 2)
plot(S5.Period,S5.fyK/9.81, 'c-', 'LineWidth', 2)
hold off
% Finalize figure
grid on
xlabel('T (sec)')
ylabel('fy/w (-)')
legend({'\mu=1','\mu=1.5','\mu=2','\mu=4','\mu=8'})
xlim([0,3])
ylim([0,1])
drawnow;
pause(0.1)
```



Copyright

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

Published with MATLAB® R2022b

verification Constant strength response spectrum

Contents

Reference

Tena-Colunga, A. (1999). Simplified seismic evaluation of existing structures, 8th Canadian Conference on Earthquake Engineering, Vancouver, Canada, 317-322.

Description

In the above reference, a new type of spectrum is proposed, the constant strength response spectrum (CSRS). A displacement ductility demand spectrum (DDDS) relates peak displacement ductility demands (and other important response quantities, i.e., displacements) with structural periods of nonlinear elastic-perfectly-plastic hysteretic SDOF systems with given yield strengths. Figure 1 of the above reference is verified in this example. The SDOF system has a yield strength ratio equal to V/W=0.15 and the acceleration time history of the SCT-EW component recorded during the 1985 Michoacan earthquake is considered.

Load earthquake data

Earthquake acceleration time history of the 1985 Michoacan earthquake will be used (SCT-EW component). The txt file contains the following columns: time, acceleration N-S, acceleration E-W, acceleration V

```
fid=fopen('sct190985.txt','r');
text=textscan(fid,'%f %f %f %f');
fclose(fid);
t=text{1,1};
dt=t(2)-t(1);
xgtt=9.81*text{1,3};
```

Calculate constant strength response spectra of earthquake motion

Switch

```
sw='csrs';
```

Eigenperiods

```
T=linspace(0.1,5,50);
```

Critical damping ratio

```
ksi=0.05;
```

Strength ratio

```
fyR=0.15;
```

Post-yield stiffness factor

```
pysf=0.001;
```

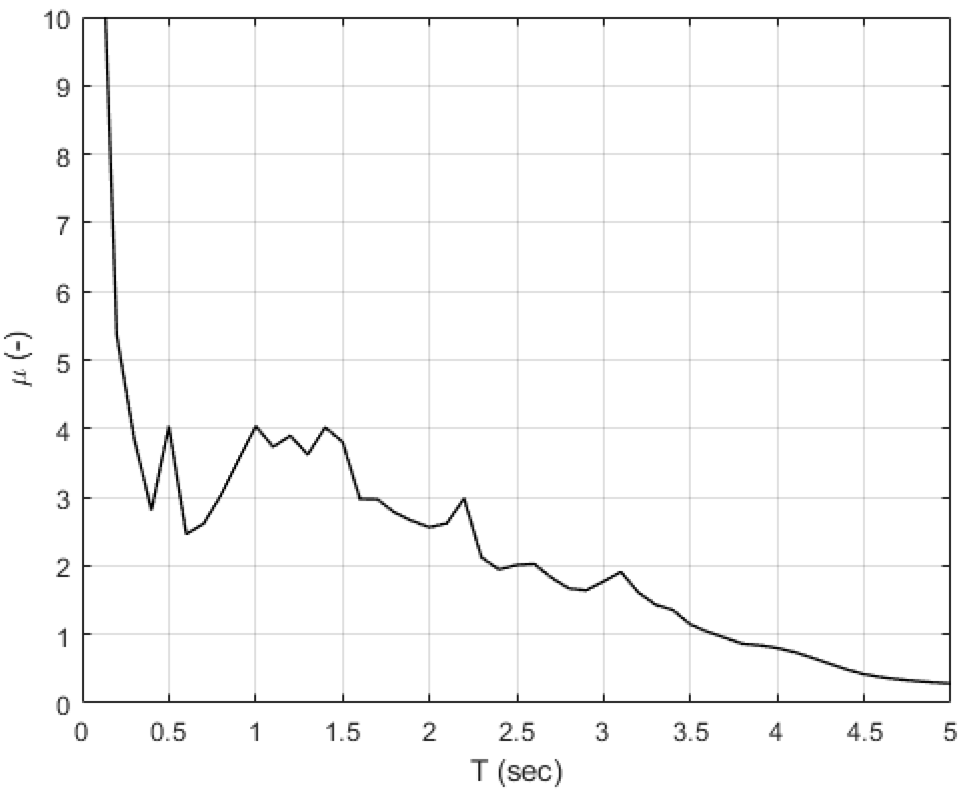
Apply OpenSeismoMatlab

```
S1=OpenSeismoMatlab(dt,xgtt,sw,T,ksi,fyR,pysf);
```

Plot the ductility demand response spectra

Initialize figure

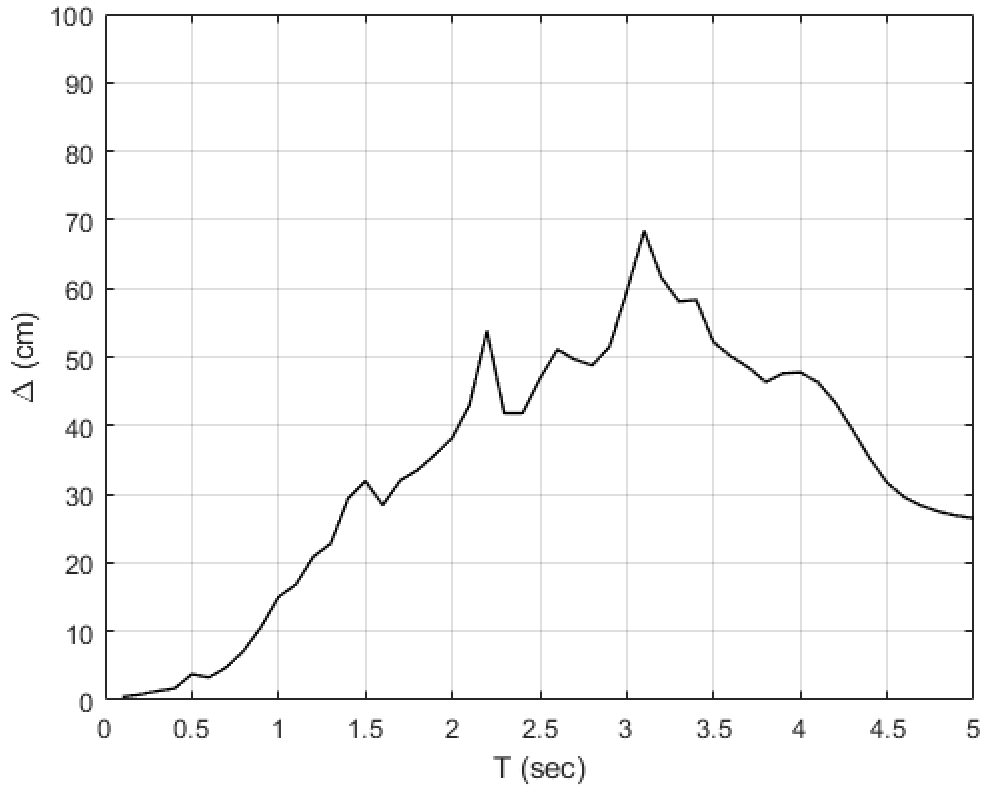
```
figure()
% Plot the constant strength ductility demand response spectra
plot(S1.Period,S1.CSSmu, 'k-', 'LineWidth', 1)
% Finalize figure
grid on
xlabel('T (sec)')
ylabel('\mu (-)')
xlim([0,5])
ylim([0,10])
drawnow;
pause(0.1)
```



Plot the displacement demand response spectra

Initialize figure

```
figure()
% Plot the constant strength displacement demand response spectra
plot(S1.Period,100*S1.CSSd, 'k-', 'LineWidth', 1)
% Finalize figure
grid on
xlabel('T (sec)')
ylabel('\Delta (cm)')
xlim([0,5])
ylim([0,100])
drawnow;
pause(0.1)
```



Copyright

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

Published with MATLAB® R2022b

verification Constant strength response spectrum

Contents

Reference

Tena-Colunga, A. "Displacement ductility demand spectra for the seismic evaluation of structures." Engineering Structures 23, no. 10 (2001): 1319-1330.

Description

The above reference presents an integral method for the seismic evaluation of the existing structures, using what the author defines as displacement ductility demand spectrum (DDDS). This DDDS is equivalent to the constant strength response spectrum (CSRS). Figure 1 of the above reference is verified in this example. The SDOF system has a yield strength ratio equal to V/W=0.10 and the acceleration time history of the SCT-EW component recorded during the 1985 Michoacan earthquake is considered.

Load earthquake data

Earthquake acceleration time history of the 1985 Michoacan earthquake will be used (SCT-EW component). The txt file contains the following columns: time, acceleration N-S, acceleration E-W, acceleration V

```
fid=fopen('sct190985.txt','r');
text=textscan(fid,'%f %f %f %f');
fclose(fid);
t=text{1,1};
dt=t(2)-t(1);
xgtt=9.81*text{1,3};
```

Calculate constant strength response spectra of earthquake motion

Switch

```
sw='csrs';
```

Eigenperiods

```
T=linspace(0.05,5,50);
```

Critical damping ratio

```
ksi=0.05;
```

Strength ratio

```
fyR=0.10;
```

Post-yield stiffness factor

```
pysf=0.001;
```

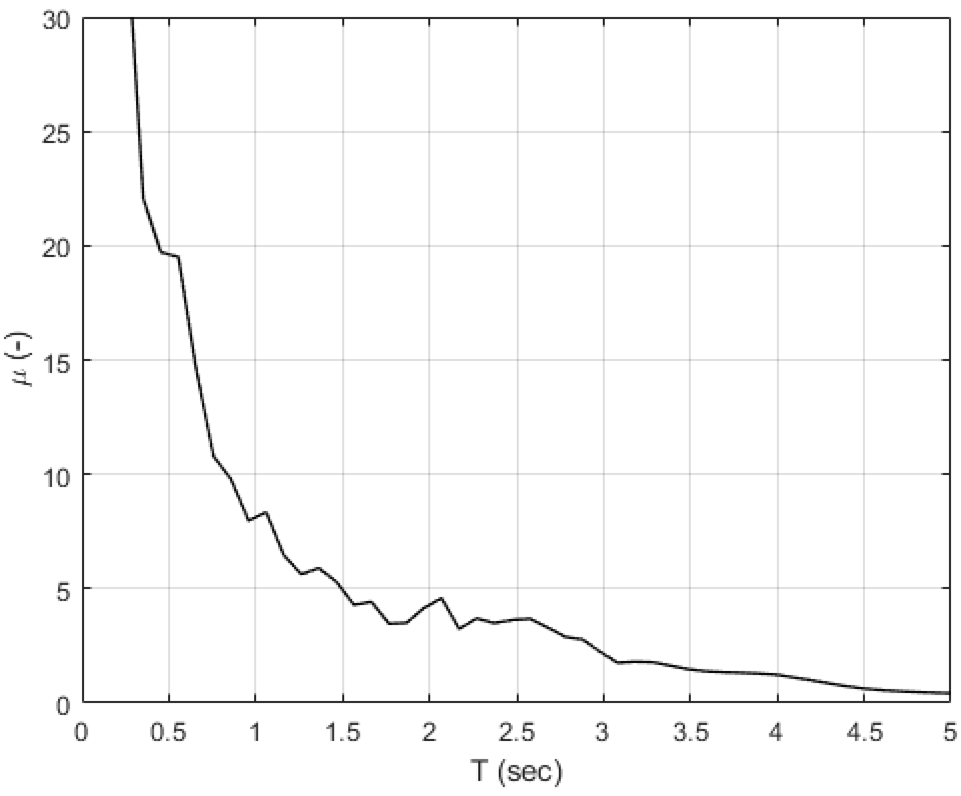
Apply OpenSeismoMatlab

```
S1=OpenSeismoMatlab(dt,xgtt,sw,T,ksi,fyR,pysf);
```

Plot the ductility demand response spectra

Initialize figure

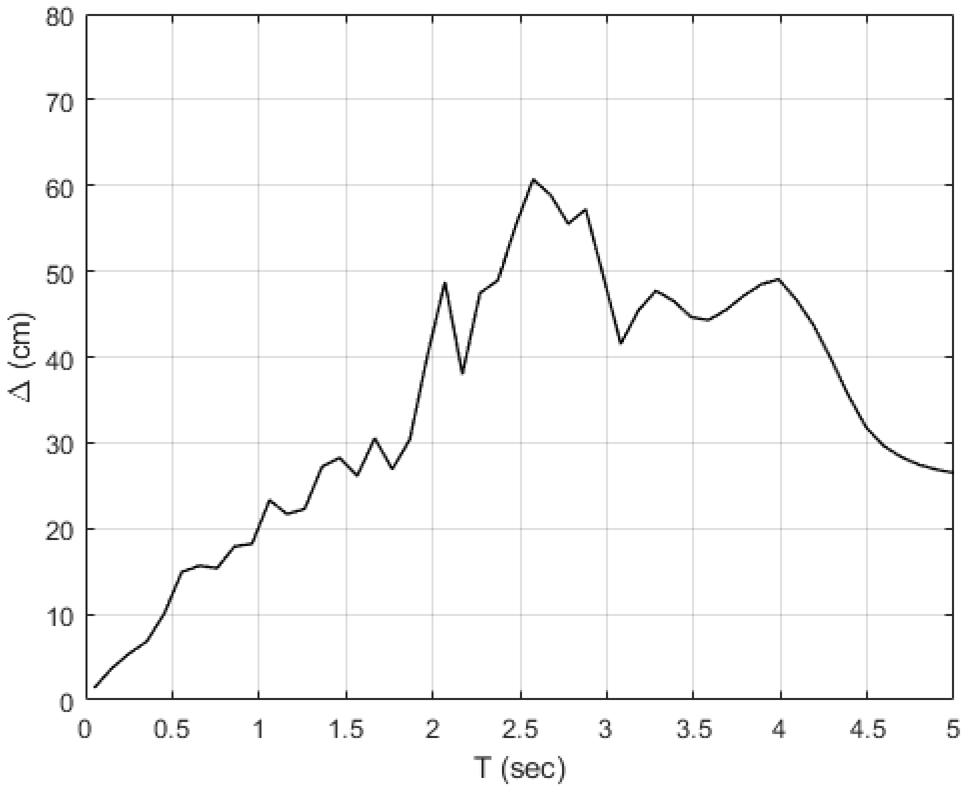
```
figure()
% Plot the constant strength ductility demand response spectra
plot(S1.Period,S1.CSSmu, 'k-', 'LineWidth', 1)
% Finalize figure
grid on
xlabel('T (sec)')
ylabel('\mu (-)')
xlim([0,5])
ylim([0,30])
drawnow;
pause(0.1)
```



Plot the displacement demand response spectra

Initialize figure

```
figure()
% Plot the constant strength displacement demand response spectra
plot(S1.Period,100*S1.CSSd, 'k-', 'LineWidth', 1)
% Finalize figure
grid on
xlabel('T (sec)')
ylabel('\Delta (cm)')
xlim([0,5])
ylim([0,80])
drawnow;
pause(0.1)
```



Copyright

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

Published with MATLAB® R2022b

verification Dynamic response history analysis of OpenSeismoMatlab

Calculate linear dynamic response of a MDOF shear building

Contents

Reference

Chopra, A. K. (2020). Dynamics of structures, Theory and Applications to Earthquake Engineering, 5th edition. Prenctice Hall.

Description

The example 13.2.6 (Example: Five-Story Shear Frame) of the above reference is solved in this example. Consider the five-story shear frame of Fig.12.8.1 of the above reference, subjected to the El Centro ground motion. The lumped masses are equal to 45 Mg at each floor, the lateral stiffness of each story is 54.82 kN/cm and the height of each story is 4 m. The damping ratio for all natural modes is 0.05.

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

Setup parameters for DRHA function

Set the storey height of the structure in m.

```
h=4;
```

Set the number of degrees of freedom of the structure, which is equal to the number of its storeys.

```
nDOFs=5;
```

Set the lateral stiffness of each storey in N/m.

```
k=5.482e6;
```

Set the lumped mass at each floor in kg.

```
m=45e3;
```

Calculation of structural properties

Calculate the stiffness matrix of the structure in N/m.

```
K=k*ones(nDOFs,1);
```

Calculate the mass matrix of the structure.

```
M=m*ones(nDOFs,1);
```

Critical damping ratio

```
ksi=0.05;
```

Initial displacement

```
u0=zeros(nDOFs,1);
```

Initial velocity

```
ut0=zeros(nDOFs,1);
```

Algorithm to be used for the time integration

```
AlgID='U0-V0-Opt';
```

Minimum absolute value of the eigenvalues of the amplification matrix

```
rinf=1;
```

Calculate dynamic response

Apply DRHA

```
[U,V,A,f,Es,Ed] = DRHA(K,M,dt,xgtt,ksi,u0,ut0,AlgID,rinf);
```

Base shear time history

```
FBeig=sum(f,1);
```

5th storey shear time history (5th DOF)

```
Feig=f(1,:);
```

Roof displacement time history (5th DOF)

```
Ueig=U(1,:);
```

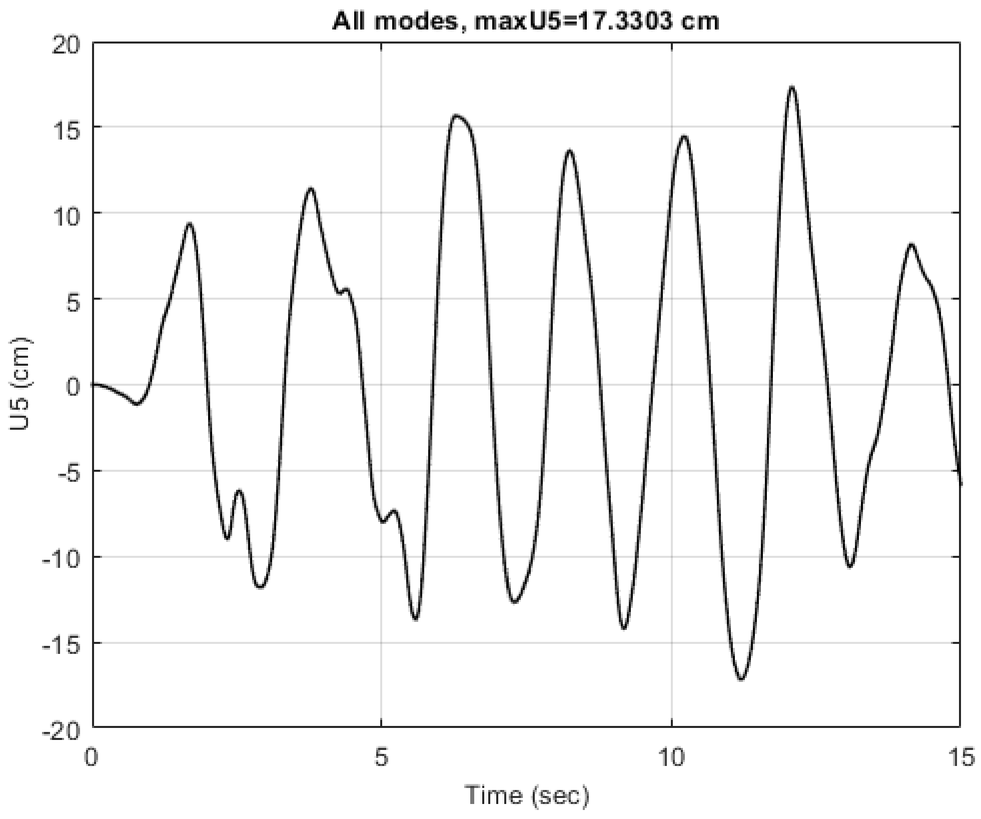
Base moment time history

```
MBeig=sum(f.*repmat((5*h:(-h):h)',1,size(f,2)),1);
```

Roof displacement time history

Plot the roof displacement time history. Convert displacements from m to cm. Verify with Figure 13.2.8 (left) of the above reference.

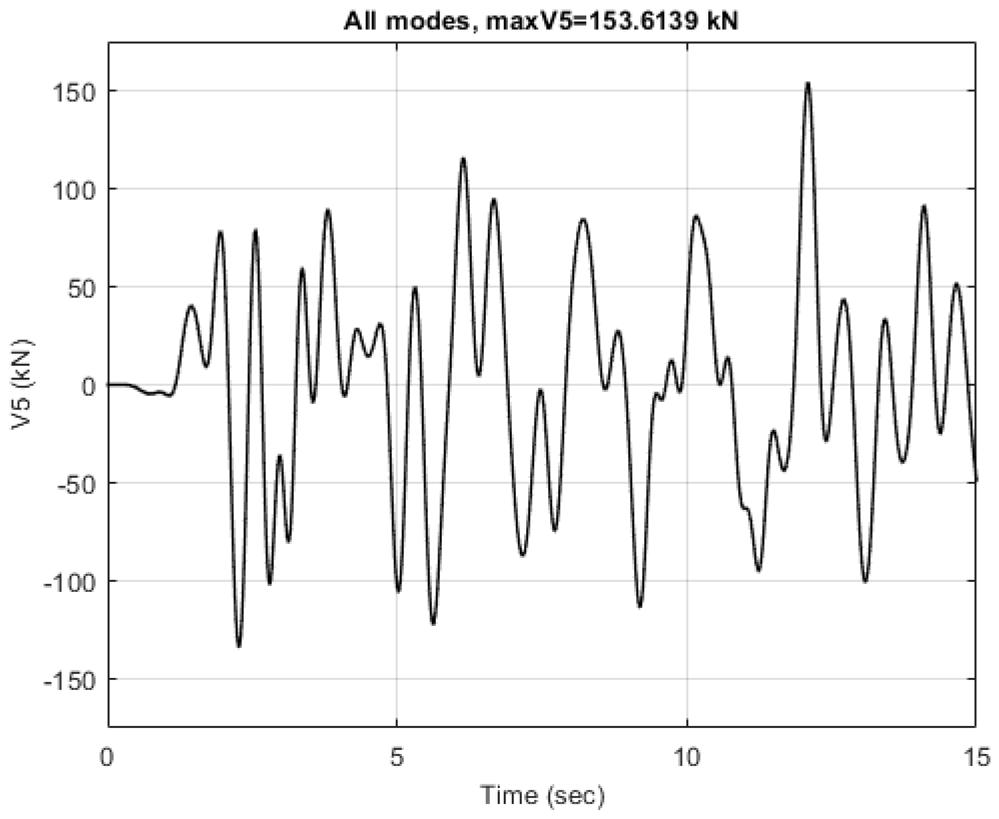
```
figure();
plot(t,100*Ueig,'LineWidth',1.,'Marker','.',...
    'MarkerSize',1,'Color',[0 0 0],'markeredgecolor','k')
grid on
xlim([0,15])
ylim([-20,20])
xlabel('Time (sec)','FontSize',10);
ylabel('U5 (cm)','FontSize',10);
title(['All modes, maxU5=',num2str(max(abs(100*Ueig))),' cm'],...
    'FontSize',10)
drawnow;
pause(0.1)
```



Fifth-story shear time history

Plot the fifth-story shear time history. Convert forces from N to kN. Verify with Figure 13.2.7 (right) of the above reference.

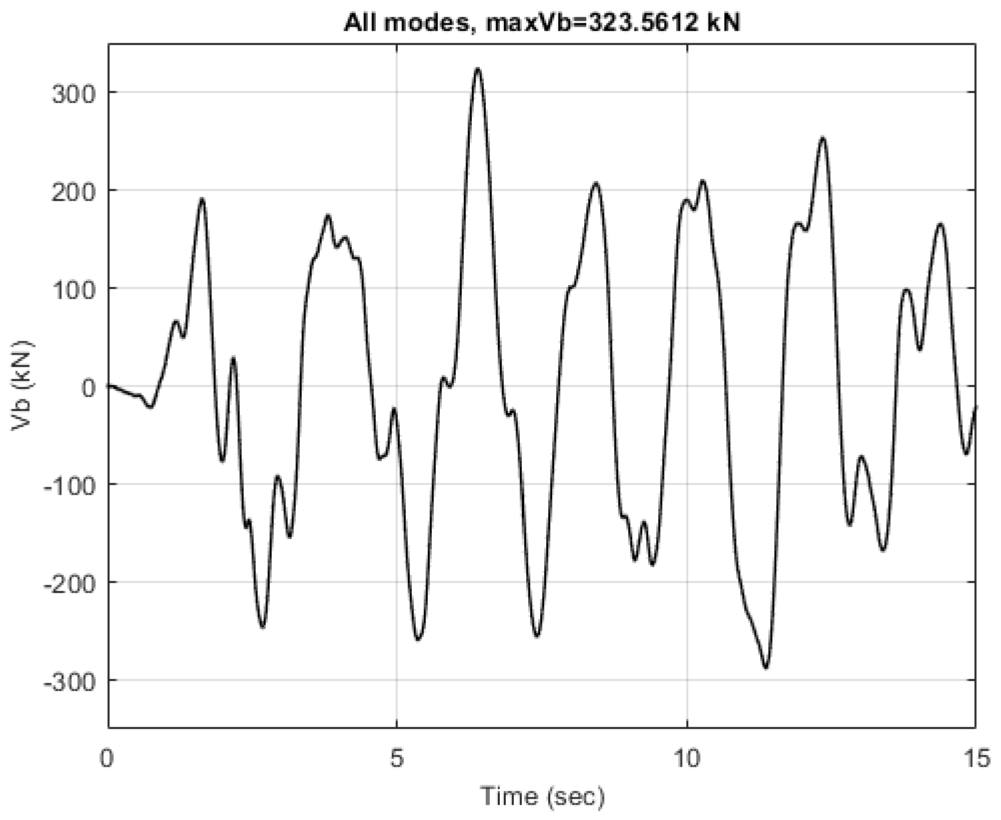
```
figure();
plot(t,Feig/1e3,'LineWidth',1.,'Marker','.',...
    'MarkerSize',1,'Color',[0 0 0],'markeredgecolor','k')
grid on
xlim([0,15])
ylim([-175,175])
xlabel('Time (sec)','FontSize',10);
ylabel('V5 (kN)','FontSize',10);
title(['All modes, maxV5=',num2str(max(abs(Feig/1e3))),' kN'],...
    'FontSize',10)
drawnow;
pause(0.1)
```



Base shear time history

Plot the base shear time history. Convert forces from N to kN. Verify with Figure 13.2.7 (left) of the above reference

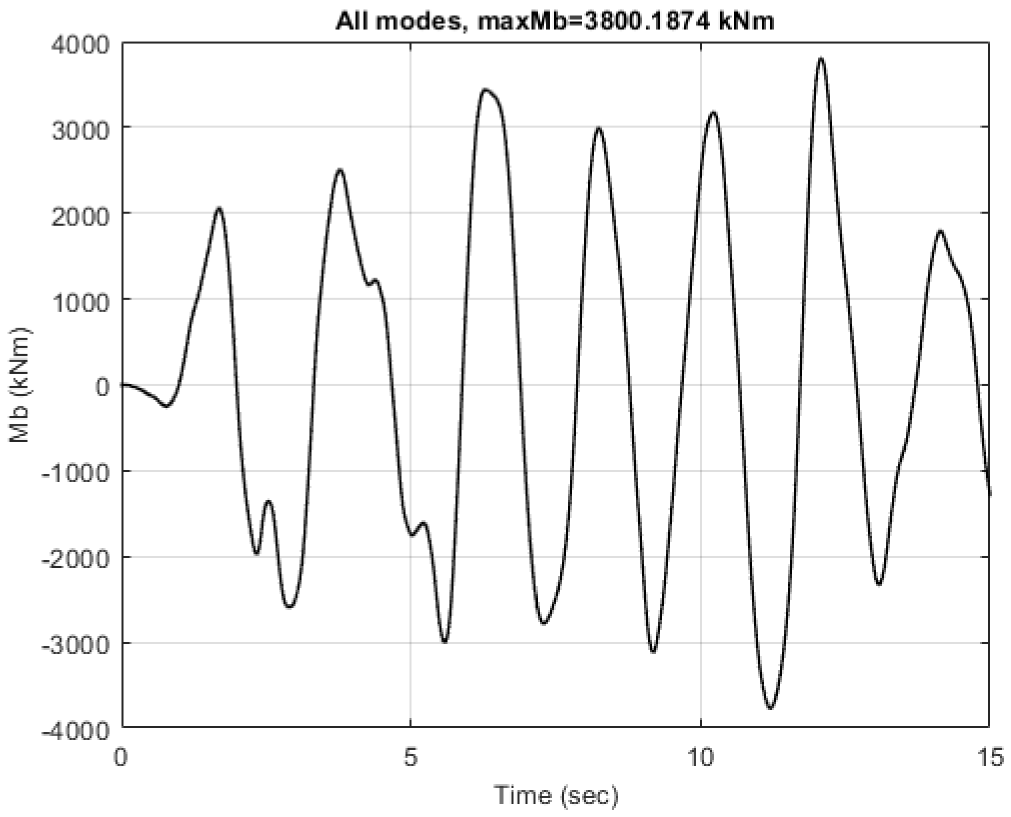
```
figure();
plot(t,FBeig/1e3,'LineWidth',1.,'Marker','.',...
    'MarkerSize',1,'Color',[0 0 0],'markeredgecolor','k')
grid on
xlim([0,15])
ylim([-350,350])
xlabel('Time (sec)','FontSize',10);
ylabel('Vb (kN)','FontSize',10);
title(['All modes, maxVb=',num2str(max(abs(FBeig/1e3))),' kN'],...
    'FontSize',10)
drawnow;
pause(0.1)
```



Base moment time history

Plot the base moment time history. Convert moments from Nm to kNm. Verify with Figure 13.2.8 (right) of the above reference

```
figure();
plot(t,MBeig/1e3,'LineWidth',1.,'Marker','.',...
    'MarkerSize',1,'Color',[0 0 0],'markeredgecolor','k')
grid on
xlim([0,15])
ylim([-4000,4000])
xlabel('Time (sec)','FontSize',10);
ylabel('Mb (kNm)','FontSize',10);
title(['All modes, maxMb=',num2str(max(abs(MBeig/1e3))),' kNm'],...
    'FontSize',10)
drawnow;
pause(0.1)
```



Copyright

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

Published with MATLAB® R2022b

verification High pass Butterworth filter of OpenSeismoMatlab

Contents

Reference

Boore, D. M. (2005). On pads and filters: Processing strong-motion data. Bulletin of the Seismological Society of America, 95(2), 745-750.

Description

Verify Figure 1 of the above reference, for the 1940 El Centro analog recording. The displacements and velocities from unfiltered and filtered accelerations are shown. Filtering is done by using two passes of a fourth-order high pass (i.e. frequencies lower than the cut-off frequency are attenuated) Butterworth filter with cut-off frequency as shown in Figure 1.

Earthquake motion

Load earthquake data

```
eqmotions={'Imperial_Valley_El_Centro_9_EW'};
data=load([eqmotions{1},'.dat']);
t=data(:,1);
dt=t(2)-t(1);
xgtt=data(:,2);
```

Apply high pass Butterworth filter

Switch

```
sw='butterworthhigh';
```

Order of Butterworth filter

```
bOrder=4;
```

Cut-off frequency

```
flc=0.1;
```

Apply OpenSeismoMatlab

```
S1=OpenSeismoMatlab(dt,xgtt,sw,bOrder,flc);
```

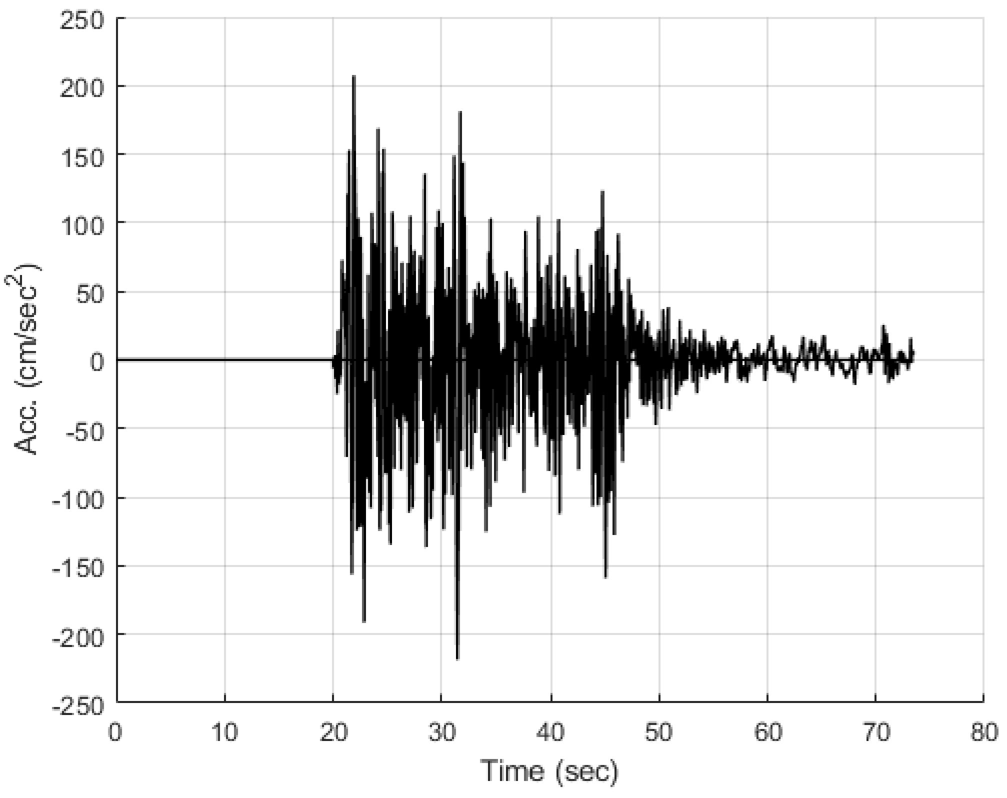
Filtered acceleration

```
cxgtt=S1.acc;
```

Plot the time history of the initial ground motion

Initialize figure

```
figure()
hold on
plot(t,zeros(size(t)),'k','LineWidth',1)
% Plot the acceleration time history of the initial ground motion
plot(t,xgtt,'k','LineWidth',1)
% Finalize figure
hold off
grid on
xlabel('Time (sec)')
ylabel('Acc. (cm/sec^2)')
drawnow;
pause(0.1)
```



Obtain displacement and velocity time histories

Switch

```
sw='timehist';
```

Do not use baseline correction

```
baselineSw=false;
```

Apply OpenSeismoMatlab to the initial ground motion

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

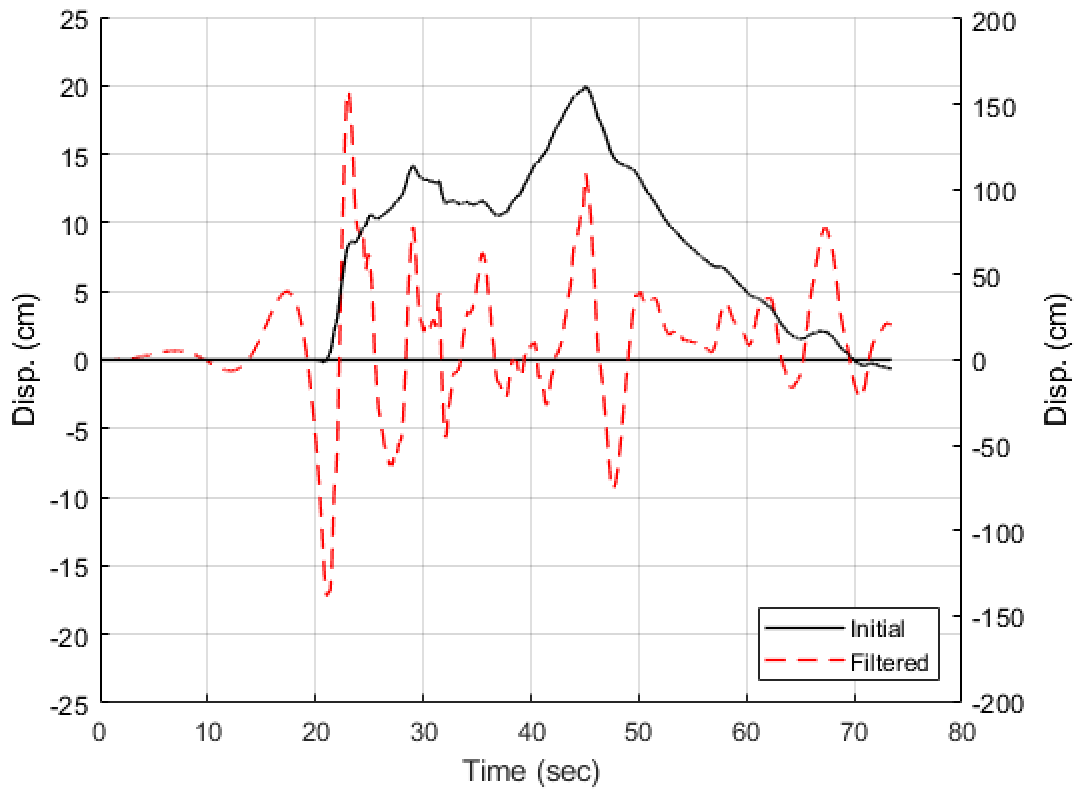
Apply OpenSeismoMatlab to the filtered ground motion

```
S3=OpenSeismoMatlab(dt,cxgtt,sw,baselineSw);
```

Plot the displacement time histories

Initialize figure

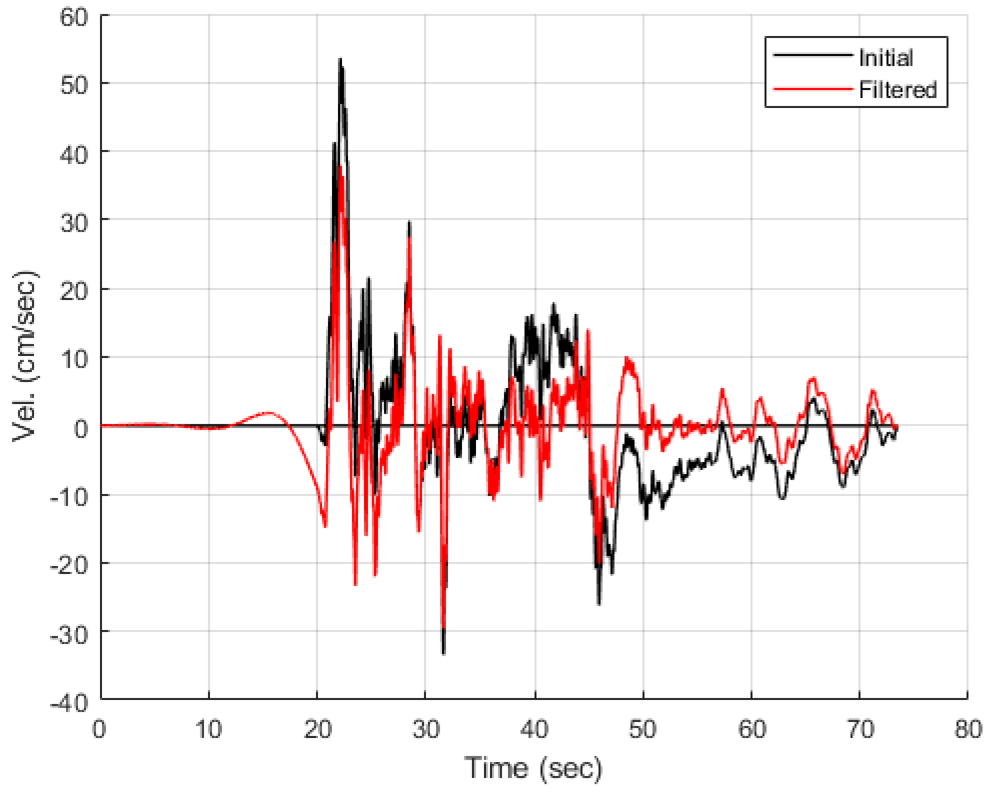
```
figure()
hold on
yyaxis left;
plot(S3.time,zeros(size(S3.time)),'k','LineWidth',1)
% Plot the displacement time history of the filtered ground motion
p2=plot(S3.time,S3.disp,'r','LineWidth',1);
ylabel('Disp. (cm)')
ylim([-25,25])
ax1 = gca;
ax1.YColor = 'k';
% Plot the displacement time history of the initial ground motion
yyaxis right;
p1=plot(S2.time,S2.disp,'k','LineWidth',1);
ylabel('Disp. (cm)')
ylim([-201,201])
ax2 = gca;
ax2.YColor = 'k';
% Finalize figure
hold off
grid on
legend([p1,p2],{'Initial','Filtered'},'location','southeast')
xlabel('Time (sec)')
drawnow;
pause(0.1)
```



Plot the velocity time histories

Initialize figure

```
figure()
hold on
plot(S3.time,zeros(size(S3.time)),'k','LineWidth',1)
% Plot the velocity time history of the initial ground motion
p1=plot(S2.time,S2.vel,'k','LineWidth',1);
% Plot the velocity time history of the filtered ground motion
p2=plot(S3.time,S3.vel,'r','LineWidth',1);
% Finalize figure
hold off
grid on
legend([p1,p2],{'Initial','Filtered'})
xlabel('Time (sec)')
ylabel('Vel. (cm/sec)')
drawnow;
pause(0.1)
```


Initial
Filtered
Vel. (cm/sec)
Time (sec)
60
50
40
30
20
10
0
-10
-20
-30
-40
0
10
20
30
40
50
60
70
80

verification Low- and high- pass Butterworth filter of OpenSeismoMatlab

Contents

Reference

Graizer, V. (2012, September). Effect of low-pass filtering and re-sampling on spectral and peak ground acceleration in strong-motion records. In Proceedings of the 15th World Conference of Earthquake Engineering, Lisbon, Portugal (pp. 24-28).

Description

Verify Figure 3.2 of the above reference, for the MW 6.3 Christchurch, New Zealand earthquake at Heathcote Valley Primary School (HVSC) station, Up-component. The time histories, elastic response spectra and Fourier spectra from unfiltered and filtered accelerations are shown and compared. In the above reference, the ground motion was processed following the 1970s Caltech procedure, low-pass filtered and re-sampled to 50 samples/sec by the GeoNet New Zealand strong motion network. However in this example, Butterworth filter is applied and it gives similar results.

Earthquake motion

Load earthquake data

```
eqmotions={'Christchurch2011HVPS_UP'};
data=load([eqmotions{1},'.dat']);
t=data(:,1);
dt=t(2)-t(1);
xgtt=data(:,2);
xgtt=[zeros(10/dt,1);xgtt];
t=(0:numel(xgtt)-1)'*dt;
xgtt=xgtt/9.81;
```

Apply high pass Butterworth filter

Switch

```
sw='butterworthhigh';
```

Order of Butterworth filter

```
bOrder=4;
```

Cut-off frequency

```
flc=0.1;
```

Apply OpenSeismoMatlab

```
S1=OpenSeismoMatlab(dt,xgtt,sw,bOrder,flc);
```

Filtered acceleration

```
cxgtt=S1.acc;
```

Apply low pass Butterworth filter

Switch

```
sw='butterworthlow';
```

Order of Butterworth filter

```
bOrder=4;
```

Cut-off frequency

```
fuc=25;
```

Apply OpenSeismoMatlab to high pass filtered filtered acceleration

```
S2=OpenSeismoMatlab(dt,cxgtt,sw,bOrder,fuc);
```

Filtered acceleration

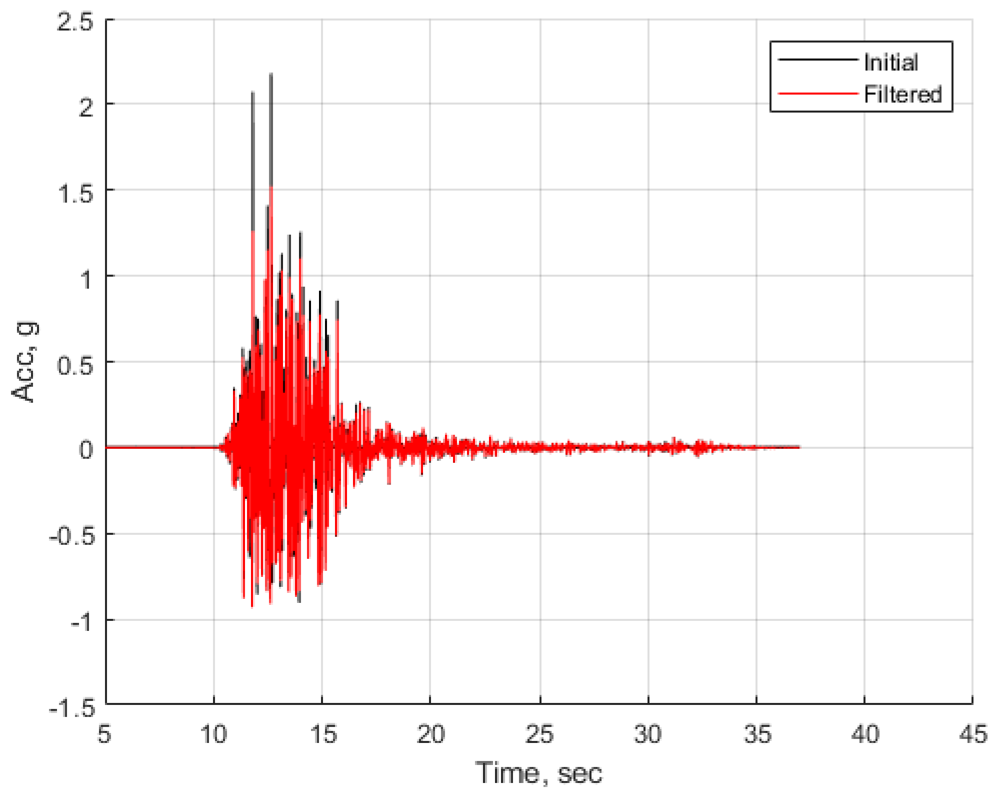
```
cxgtt=S2.acc;
```

Plot the acceleration time histories

Initialize figure

```
figure()
hold on
plot(t,zeros(size(t)),'k','LineWidth',1)
```

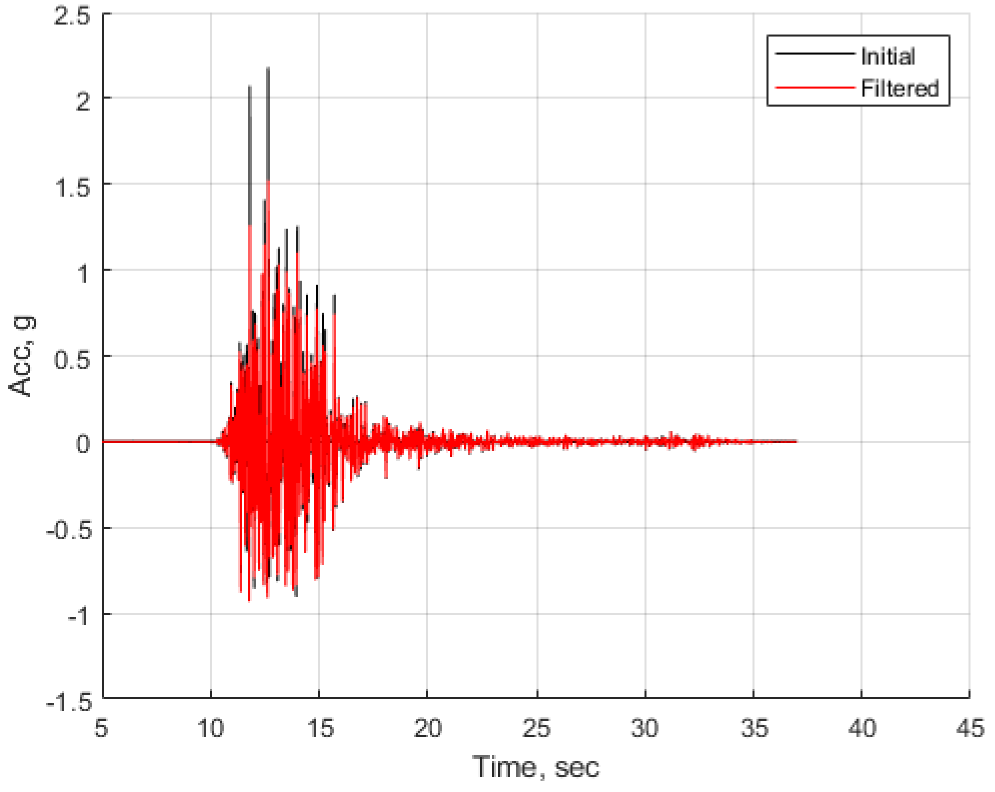
```
% Plot the acceleration time history of the initial ground motion
p1=plot(t,xgtt,'k');
% Plot the acceleration time history of the bandpass filtered ground motion
p2=plot(t,cxgtt,'r');
% Finalize figure
hold off
grid on
legend([p1,p2],{'Initial','Filtered'})
xlabel('Time, sec')
ylabel('Acc, g')
xlim([5,45])
ylim([-1.5,2.5])
drawnow;
pause(0.1)
```

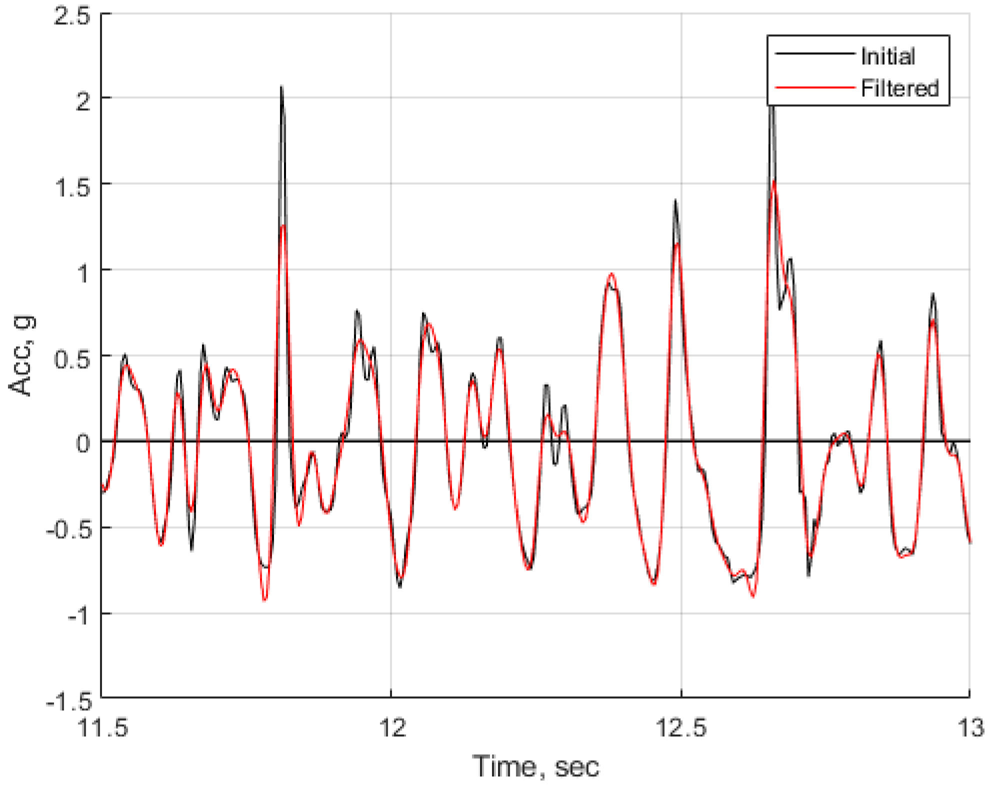


Plot a zoom-in view of the acceleration time histories

Capture the handle of the plot of the acceleration time histories

```
originalFigure = gcf;
% Create a copy of the figure
copiedFigure = copyobj(originalFigure, 0);
% Show the copied figure
figure(copiedFigure);
% Apply zoom along x-axis
xlim([11.5,13])
drawnow;
pause(0.1)
```





Calculate the Fourier spectra

Switch

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

Apply OpenSeismoMatlab to the initial ground motion

```
S3=OpenSeismoMatlab(dt,xgtt,sw);
```

Apply OpenSeismoMatlab to the filtered ground motion

```
S4=OpenSeismoMatlab(dt,cxgtt,sw);
```

Plot the Fourier spectra

Initialize figure

```
figure()
loglog(1,1,'w')
hold on
% Plot the Fourier spectrum of the initial ground motion in logarithmic
% scale for frequencies larger than 0.05 Hz
ind10=S4.freq>=0.05;
p1=loglog(S3.freq(ind10),S3.FAS(ind10),'k','LineWidth',1);
% Plot the Fourier spectrum of the filtered ground motion in logarithmic
% scale for frequencies larger than 0.05 Hz and lower than fuc
ind11=(S4.freq<=fuc)& (S4.freq>=0.05);
p2=loglog(S4.freq(ind11),S4.FAS(ind11),'r','LineWidth',1);
% Finalize figure
hold off
grid on
xlim([1e-2,1e2])
ylim([1e-4,1])
legend([p1,p2],{'Initial','Filtered'})
xlabel('Frequency, Hz')
ylabel('Fourier spectra, g*sec')
drawnow;
pause(0.1)
```

Calculate the acceleration response spectra

Switch

```
sw='elrs';
```

Critical damping ratio

```
ksi=0.05;
% Period range for which the response spectrum is queried
T=logspace(1,-2,100);
```

Apply OpenSeismoMatlab to the initial ground motion

```
S5=OpenSeismoMatlab(dt,xgtt,sw,T,ksi);
```

Apply OpenSeismoMatlab to the filtered ground motion

```
S6=OpenSeismoMatlab(dt,cxgtt,sw,T,ksi);
```

Plot the acceleration response spectra

Initialize figure

```
figure()
semilogx(1,1,'w')
hold on
% Plot the acceleration response spectrum of the initial ground motion in
% logarithmic scale
p1=semilogx(1./S5.Period,S5.Sa,'k','LineWidth',1);
% Plot the acceleration response spectrum of the filtered ground motion in
% logarithmic scale
p2=semilogx(1./S6.Period,S6.Sa,'r','LineWidth',1);
% Finalize figure
hold off
grid on
legend([p1,p2],{'Initial','Filtered'})
xlabel('Frequency, Hz')
ylabel('Spectral acceleration, g')
xlim([0.1,100])
ylim([0,6])
drawnow;
pause(0.1)
```

Copyright

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

Published with MATLAB® R2022b

verification Fourier amplitude spectrum of OpenSeismoMatlab

Contents

Reference

Analyses of strong motion earthquake accelerograms, Volume IV - Fourier Amplitude Spectra, Part H - Accelerograms IIH115 through IIH126, California Institute of Technology, Earthquake Engineering Research Laboratory, Report No. EERI 74-100, 1974.

Description

Verify the Fourier amplitude spectrum at page 12 of the above reference for the San Fernando earthquake, Feb 9, 1971, 0600 PST, IVH115 71.024.0 15250 Ventura BLVD., basement, Los Angeles, Cal. Component N11E.

Earthquake motion

Load earthquake data

```
eqmotions={'SanFernando1971VenturaBlvdBasement15250LosAngelesCalN11E'};
data=load([eqmotions{1},'.dat']);
t=data(:,1);
dt=t(2)-t(1);
xgtt=100*data(:,2);
```

Calculate the Fourier amplitude spectrum

Switch

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

Apply OpenSeismoMatlab

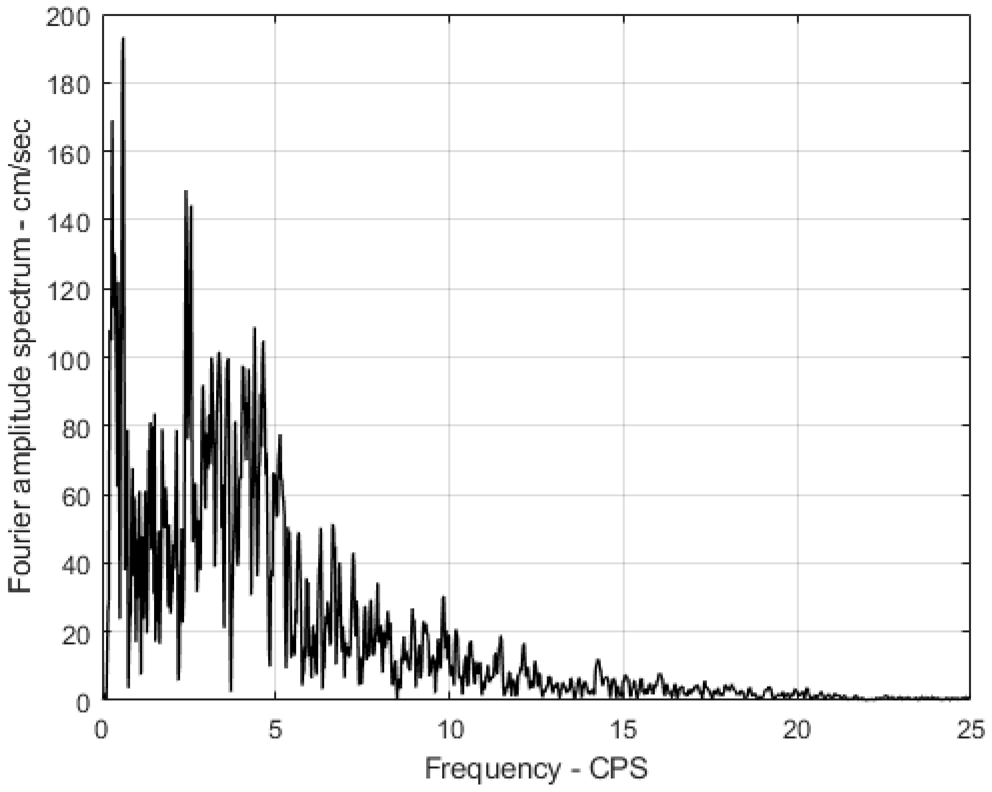
```
S1=OpenSeismoMatlab(dt,xgtt,sw);
```

Plot the Fourier amplitude spectrum

Initialize figure

```
figure()
% Plot the Fourier amplitude spectrum on page 12 of the above reference
plot(S1.freq,S1.FAS,'k','LineWidth',1)
% Finalize figure
```

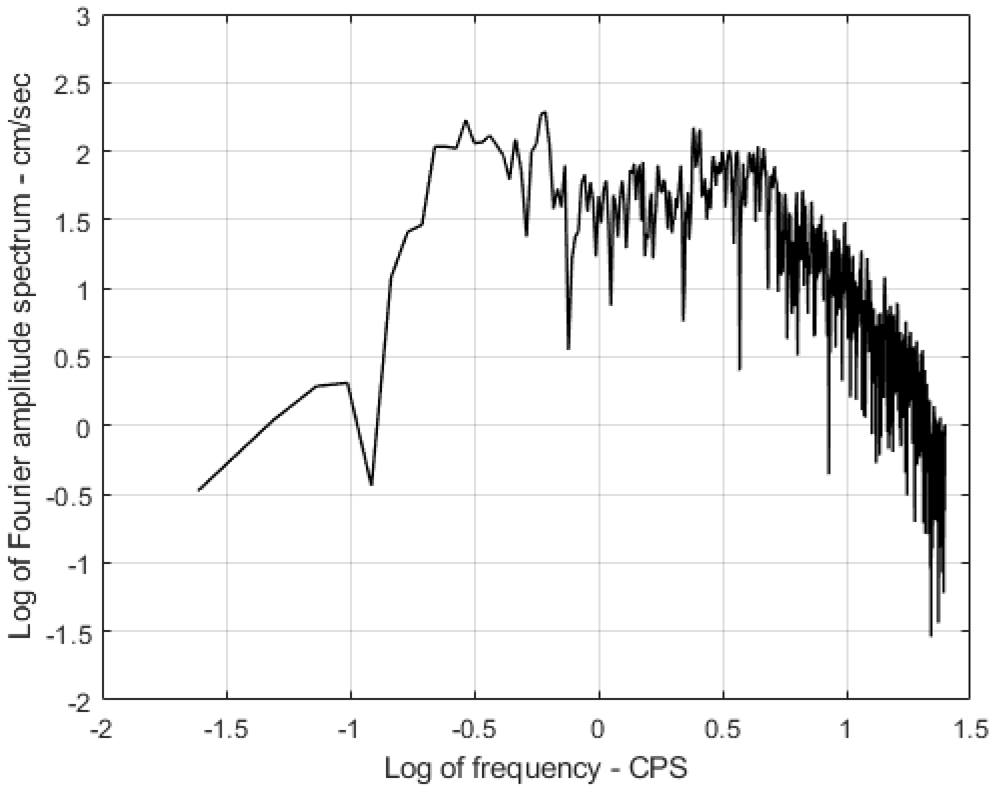
```
grid on
xlabel('Frequency - CPS')
ylabel('Fourier amplitude spectrum - cm/sec')
ylim([0,200])
xlim([0,25])
drawnow;
pause(0.1)
```



Plot the Fourier amplitude spectrum in logarithmic scale

Initialize figure

```
figure()
% Plot the Fourier amplitude spectrum on page 13 of the above reference
plot(log10(S1.freq),log10(S1.FAS),'k','LineWidth',1)
% Finalize figure
grid on
xlabel('Log of frequency - CPS')
ylabel('Log of Fourier amplitude spectrum - cm/sec')
ylim([-2,3])
xlim([-2,1.5])
drawnow;
pause(0.1)
```



Copyright

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

Published with MATLAB® R2022b

verification Fourier amplitude spectrum of OpenSeismoMatlab

Contents

Reference

Analyses of strong motion earthquake accelerograms, Volume IV - Fourier Amplitude Spectra, Part H - Accelerograms IIH115 through IIH126, California Institute of Technology, Earthquake Engineering Research Laboratory, Report No. EERl 74-100, 1974.

Description

Verify the Fourier amplitude spectrum at page 14 of the above reference for the San Fernando earthquake, Feb 9, 1971, 0600 PST, IVH115 71.024.0 15250 Ventura BLVD., basement, Los Angeles, Cal. Component N79W.

Earthquake motion

Load earthquake data

```
eqmotions={'SanFernando1971VenturaBlvdBasement15250LosAngelesCalN79W'};
data=load([eqmotions{1},'.dat']);
t=data(:,1);
dt=t(2)-t(1);
xgtt=100*data(:,2);
```

Calculate the Fourier amplitude spectrum

Switch

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

Apply OpenSeismoMatlab

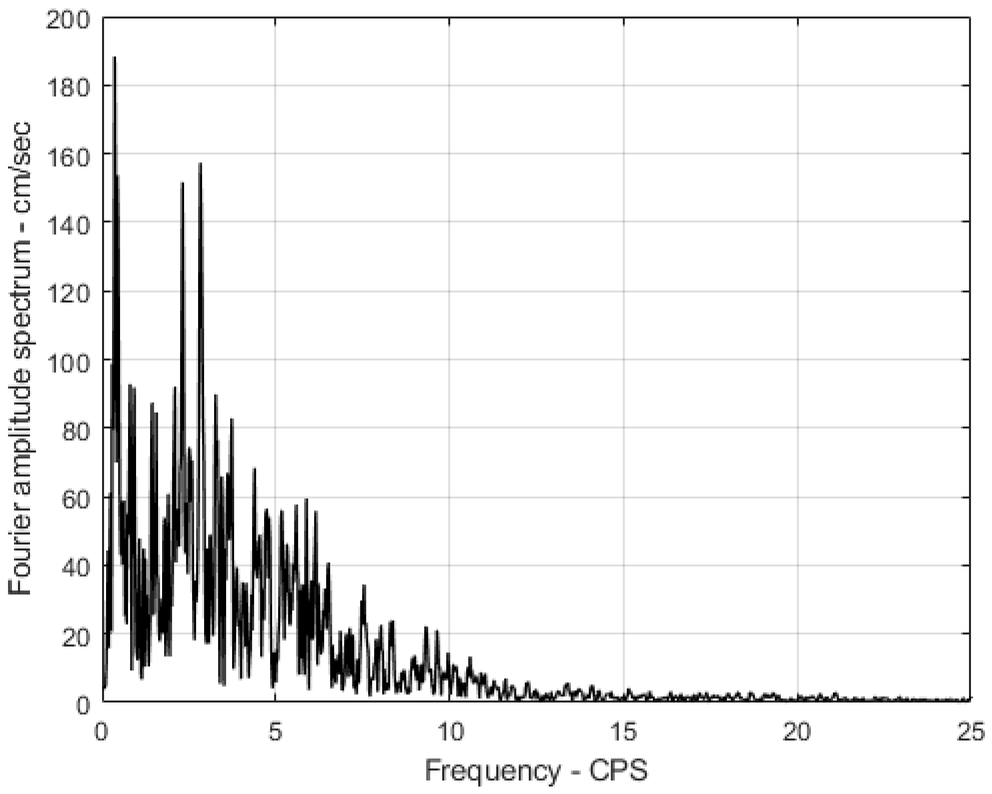
```
S1=OpenSeismoMatlab(dt,xgtt,sw);
```

Plot the Fourier amplitude spectrum

Initialize figure

```
figure()
% Plot the Fourier amplitude spectrum on page 14 of the above reference
plot(S1.freq,S1.FAS,'k','LineWidth',1)
% Finalize figure
```

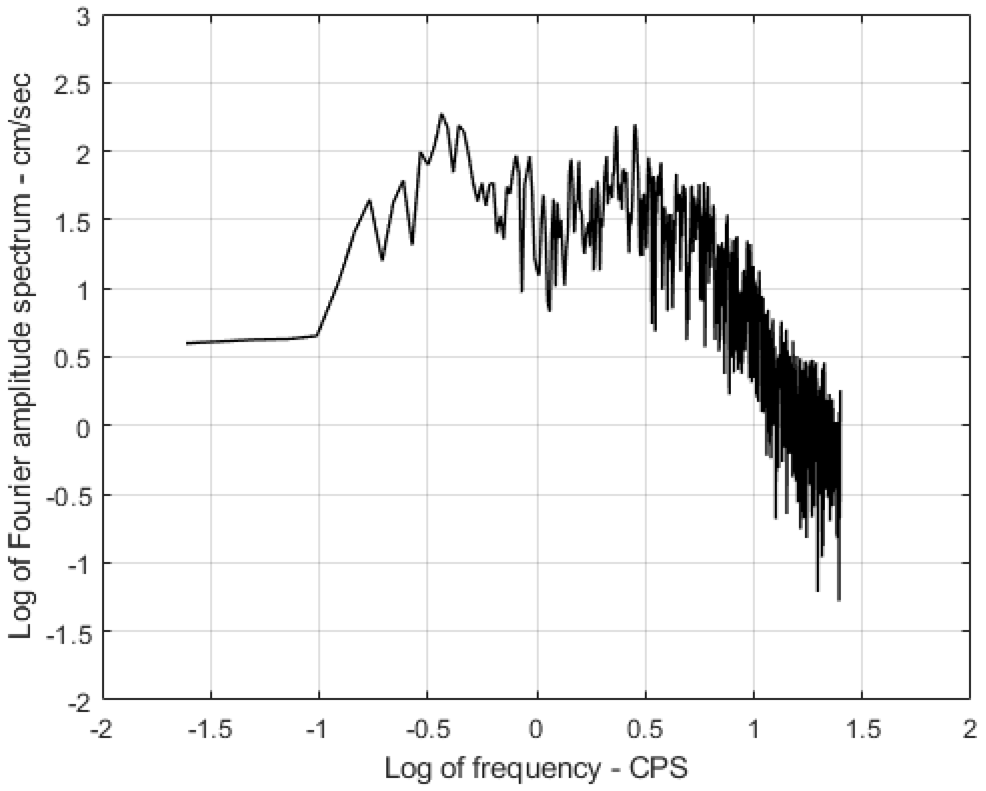
```
grid on
xlabel('Frequency - CPS')
ylabel('Fourier amplitude spectrum - cm/sec')
ylim([0,200])
xlim([0,25])
drawnow;
pause(0.1)
```



Plot the Fourier amplitude spectrum in logarithmic scale

Initialize figure

```
figure()
% Plot the Fourier amplitude spectrum on page 15 of the above reference
plot(log10(S1.freq),log10(S1.FAS),'k','LineWidth',1)
% Finalize figure
grid on
xlabel('Log of frequency - CPS')
ylabel('Log of Fourier amplitude spectrum - cm/sec')
ylim([-2,3])
xlim([-2,2])
drawnow;
pause(0.1)
```


Log of Fourier amplitude spectrum - cm/sec
Log of frequency - CPS
3
2.5
2
1.5
1
0.5
0
-0.5
-1
-1.5
-2
-2
-1.5
-1
-0.5
0
0.5
1
1.5
2

verification Incremental dynamic analysis of OpenSeismoMatlab

Contents

Reference

Mashayekhi, M., Harati, M., Darzi, A., & Estekanchi, H. E. (2020). Incorporation of strong motion duration in incremental-based seismic assessments. Engineering Structures, 223, 111144.

Description

Incremental dynamic analysis (IDA) is performed for a non-degrading SDOF model with eigenperiod T=1 sec. The employed hysteretic model is a bilinear elastoplastic model used for non-degrading SDOF systems and is shown in Figure 17(a) of the above reference. An IDA analysis is performed with a ground motion the spectral acceleration of which resembles the red line of Figure 14 of the above reference, i.e. the ground motion must have Sa(1 sec)=0.382g (which is the Intensity Measure - IM) and the duration D_5_75 must be roughly equal to 8.3 sec. An acceleration time history with such characteristics is shown in Figure 16(c) of the above reference. In this example, an arbitrary ground motion acceleration is loaded, which is then adjusted so that the resulting time history has Sa(1 sec)=0.382g and D_5_75=8.3 sec. The adjusted time history is plotted in this example and can be compared to Figure 16(c) of the above reference. Based on the above problem statement, the median response curve of Figure 18(a) of the above reference is verified.

Earthquake motion

Load earthquake data

```
eqmotions={'LomaPrietaHallsValley90'};
data=load([eqmotions{1},'.dat']);
t=data(:,1);
dt=t(2)-t(1);
xgtt=data(:,2);
```

Adjust earthquake motion to have D_5_75=8.3sec

Switch

```
sw='arias';
```

Apply OpenSeismoMatlab

```
S1=OpenSeismoMatlab(dt,xgtt,sw);
```

Duration D_5_75 of the initially loaded motion

```
S1.Td_5_75
```

```
ans =

                       7.78
```

S.Td_5_75 must be roughly near 8.3 sec, as required in Mashayekhi et al. (2020) We manipulate the strong shaking part of the motion which corresponds to the significant duration so that S.Td_5_75 is increased to the desired value (8.3 sec)

```
id1=find(t==S1.t_5_75(1));
id2=find(t==S1.t_5_75(2));
xgtt(id1:id2)=0.8*xgtt(id1:id2);
```

Calculate duration D_5_75 of adjusted earthquake motion

Switch

```
sw='arias';
```

Apply OpenSeismoMatlab

```
S2=OpenSeismoMatlab(dt,xgtt,sw);
```

Duration D_5_75 of the adjusted motion

```
S2.Td_5_75
```

```
ans =

                       7.78
```

Scale earthquake motion to have Sa(1 sec)=0.382g

Switch

```
sw='elrs';
```

Critical damping ratio

```
ksi=0.05;
```

Period where Sa=0.382g

```
T=1;
```

Apply OpenSeismoMatlab

```
S3=OpenSeismoMatlab(dt,xgtt,sw,T,ksi);
```

Spectral acceleration of the adjusted motion at 1 sec

```
S3.Sa
```

```
ans =

          1.42095612026039
```

Sa at 1 sec must be equal to 0.382g, so we scale the entire acceleration time history up to this level

```
scaleF=0.382*9.81/S3.Sa;
xgtt=xgtt*scaleF;
```

Calculate spectral acceleration of scaled earthquake motion

Switch

```
sw='elrs';
```

Critical damping ratio

```
ksi=0.05;
```

Period where Sa=0.382g

```
T=1;
```

Apply OpenSeismoMatlab

```
S4=OpenSeismoMatlab(dt,xgtt,sw,T,ksi);
```

Spectral acceleration of the adjusted motion at 1 sec

```
S4.Sa
```

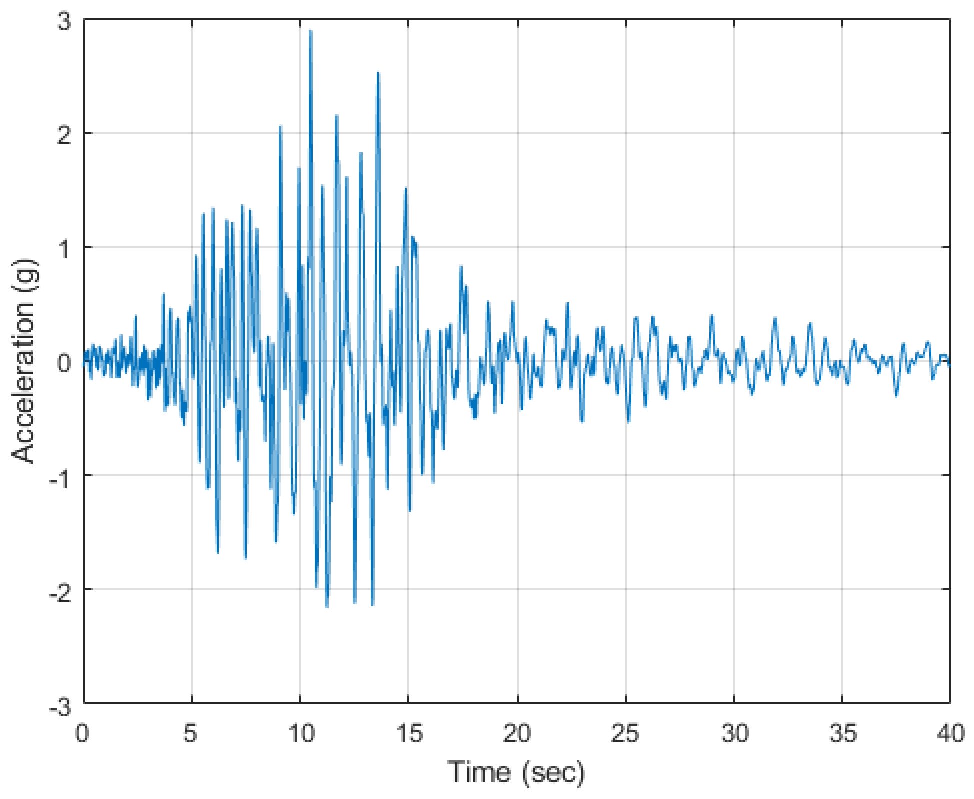
```
ans =

                  3.74742
```

Plot the acceleration time history

Initialize figure

```
figure()
% Plot the acceleration time history of the adjusted motion
plot(t,xgtt)
% Finalize figure
grid on
xlabel('Time (sec)')
ylabel('Acceleration (g)')
drawnow;
pause(0.1)
```



Perform IDA analysis

Switch

```
sw='ida';
```

Eigenperiod

```
T=1;
```

Scaling factors

```
lambdaF=logspace(log10(0.001),log10(10),100);
```

Type of IDA analysis

```
IM_DM='Sa_disp';
```

Mass

```
m=1;
```

Yield displacement

```
uy = 0.082*9.81/(2*pi/T)^2;
```

Post yield stiffness factor

```
pysf=0.01;
```

Fraction of critical viscous damping

```
ksi=0.05;
```

Apply OpenSeismoMatlab

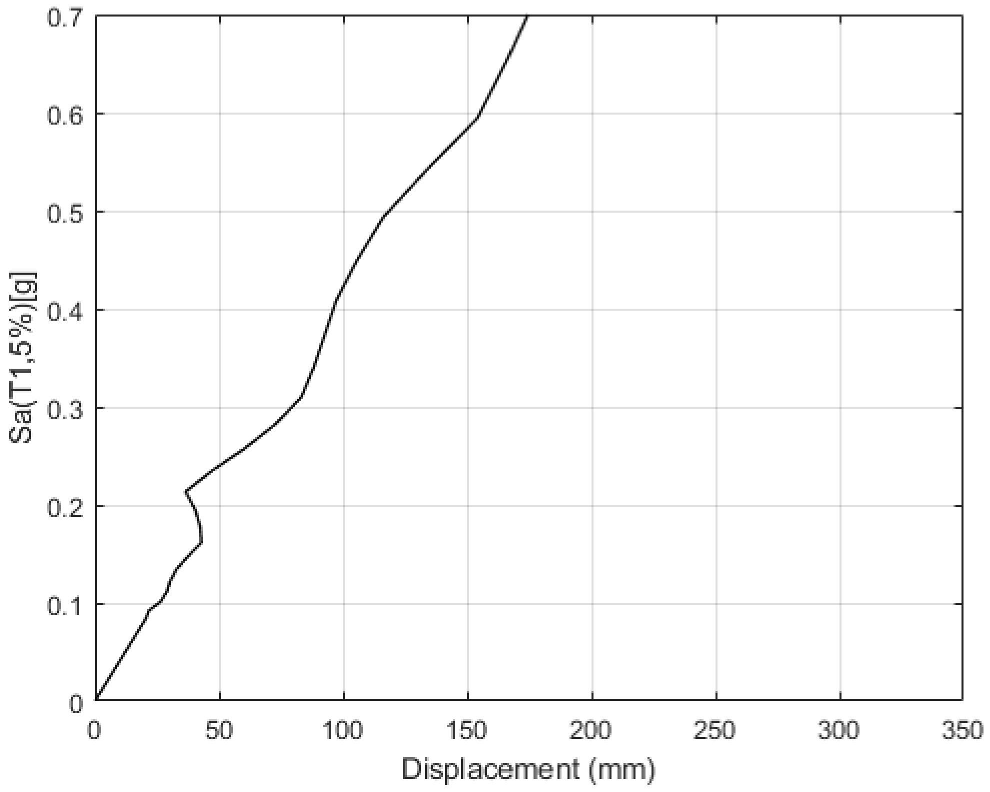
```
S5=OpenSeismoMatlab(dt,xgtt,sw,T,lambdaF,IM_DM,m,uy,pysf,ksi);
```

Plot the displacement time histories

Initialize figure

```
figure()
% Plot the response curve of the incremental dynamic analysis
plot(S5.DM*1000,S5.IM/9.81,'k','LineWidth',1)
% Finalize figure
grid on
xlabel('Displacement (mm)')
ylabel('Sa(T1,5%)[g]')
xlim([0,350])
```

```
ylim([0,0.7])
drawnow;
pause(0.1)
```



Copyright

verification Incremental dynamic analysis for multiple motions

Contents

Reference

Deng, P., Pei, S., van de Lindt, J. W., Liu, H., & Zhang, C. (2017). An approach to quantify the influence of ground motion uncertainty on elastoplastic system acceleration in incremental dynamic analysis. Advances in Structural Engineering, 20(11), 1744-1756.

Description

Figure 4(a) of the above reference contains the IDA curves of an elastoplastic SDOF system under the Consortium of Universities for Research in Earthquake Engineering (CUREE) GM suite (Krawinkler et al., 2001), which were constructed using the maximum acceleration. In this example, an arbitrary suite of strong ground motions is selected and the maximum acceleration IDA curves are constructed similar to Figure 4(a) of the above reference. The IDA curves of this example strongly resemble those of that figure.

Earthquake motions

Load data from a suite of earthquakes

```
GM={'Cape Mendocino.dat';
    'ChiChi.dat';
    'Christchurch2011HVPS_UP.dat';
    'Imperial Valley.dat';
    'Imperial_Valley_El_Centro_9_EW.dat';
    'Kobe.dat';
    'Kocaeli.dat';
    'San Fernando.dat';
    'Spitak.dat'};
n=size(GM,1);
dt=cell(n,1);
xgtt=cell(n,1);
for i=1:n
    fid=fopen(GM{i},'r');
    text=textscan(fid,'%f %f');
    fclose(fid);
    t=text{1,1};
    dt{i}=t(2)-t(1);
    xgtt{i}=text{1,2};
end
```

Setup parameters for IDA analysis

Switch

```
sw='ida';
```

Eigenperiod

```
T=1;
```

Scaling factors

```
lambdaF=logspace(log10(0.001),log10(10),100);
```

Type of IDA analysis

```
IM_DM='pgv_acc';
```

Mass

```
m=1;
```

Yield displacement

```
uy = 0.18*9.81/(2*pi/T)^2;
```

Post yield stiffness factor

```
pysf=0.01;
```

Fraction of critical viscous damping

```
ksi=0.05;
```

Algorithm to be used for the time integration

```
AlgID='U0-V0-Opt';
```

Set initial displacement

```
u0=0;
```

Set initial velocity

```
ut0=0;
```

Minimum absolute value of the eigenvalues of the amplification matrix

```
rinf=1;
```

Maximum tolerance for convergence

```
maxtol=0.01;
```

Maximum number of iterations per increment

```
jmax=200;
```

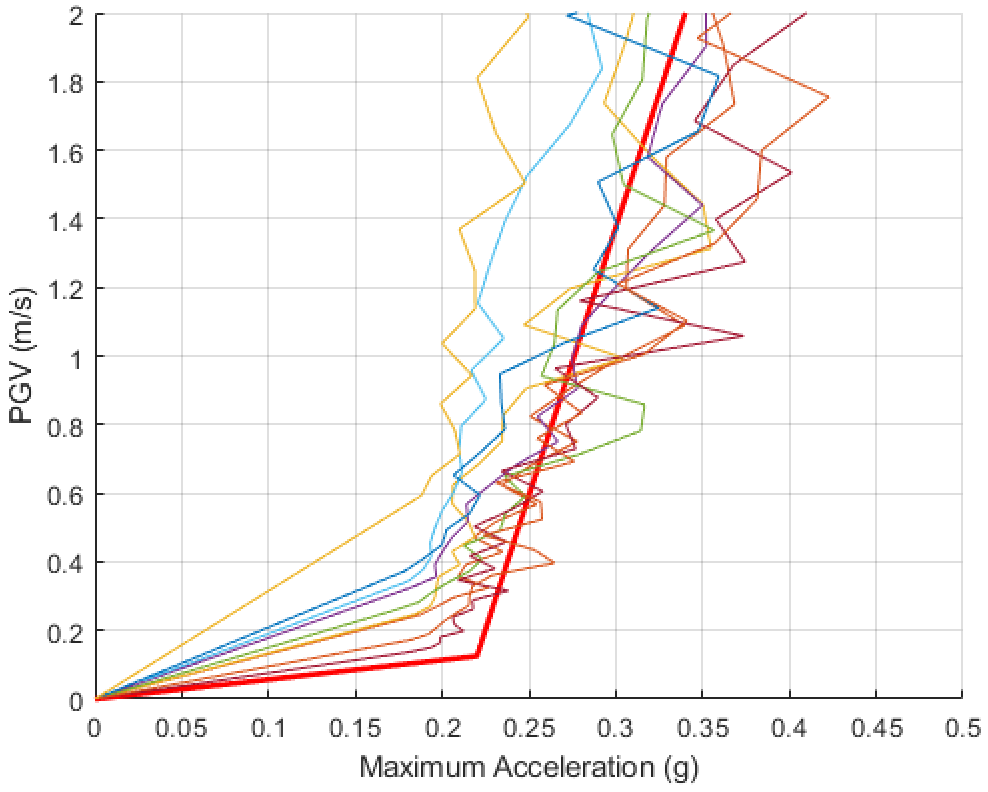
Infinitesimal variation of acceleration

```
dak=eps;
```

Construct and plot the IDA curves in a loop

Initialize figure

```
figure()
hold on
% Plot the red bold curve of Figure 4(a) of the above reference
plot([0,0.22,0.34],[0,0.125,2],'r','LineWidth',2)
for i=1:n
    % Apply OpenSeismoMatlab to calculate the ith IDA curve
    S1=OpenSeismoMatlab(dt{i},xgtt{i},sw,T,lambdaF,IM_DM,m,uy,pysf,ksi,AlgID,...
        u0,ut0,rinf,maxtol,jmax,dak);
    % Plot the ith IDA curve
    plot(S1.DM/9.81,S1.IM)
end
% Finalize figure
grid on
xlabel('Maximum Acceleration (g)')
ylabel('PGV (m/s)')
xlim([0,0.5])
ylim([0,2])
drawnow;
pause(0.1)
```



Copyright

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

Published with MATLAB® R2022b

verification Incremental dynamic analysis for ductility response

Contents

Reference

Vamvatsikos, D., & Cornell, C. A. (2002). Incremental dynamic analysis. Earthquake engineering & structural dynamics, 31(3), 491-514.

Description

The ductility response IDA curve of an elastoplastic SDOF system excited by the Loma Prieta, 1989, Halls Valley earthquake (component 090) is constructed at multiple levels of shaking, and compared to the curve shown in Figure 4(a) of the above reference. The SDOF system has T=1 sec and critical damping ratio 5%

Earthquake motion

Load earthquake data

```
GM='LomaPrietaHallsValley90.dat';
fid=fopen(GM,'r');
text=textscan(fid,'%f %f');
fclose(fid);
t=text{1,1};
dt=t(2)-t(1);
xgtt=text{1,2};
```

Perform IDA analysis

Switch

```
sw='ida';
```

Eigenperiod

```
T=1;
```

Scaling factors

```
lambdaF=logspace(log10(0.01),log10(30),100);
```

Type of IDA analysis

```
IM_DM='Sa_mu';
```

Mass

```
m=1;
```

Yield displacement

```
uy=0.25;
```

Post yield stiffness factor

```
pysf=0.01;
```

Fraction of critical viscous damping

```
ksi=0.05;
```

Algorithm to be used for the time integration

```
AlgID='U0-V0-Opt';
```

Set initial displacement

```
u0=0;
```

Set initial velocity

```
ut0=0;
```

Minimum absolute value of the eigenvalues of the amplification matrix

```
rinf=1;
```

Maximum tolerance for convergence

```
maxtol=0.01;
```

Maximum number of iterations per increment

```
jmax=200;
```

Infinitesimal variation of acceleration

```
dak=eps;
```

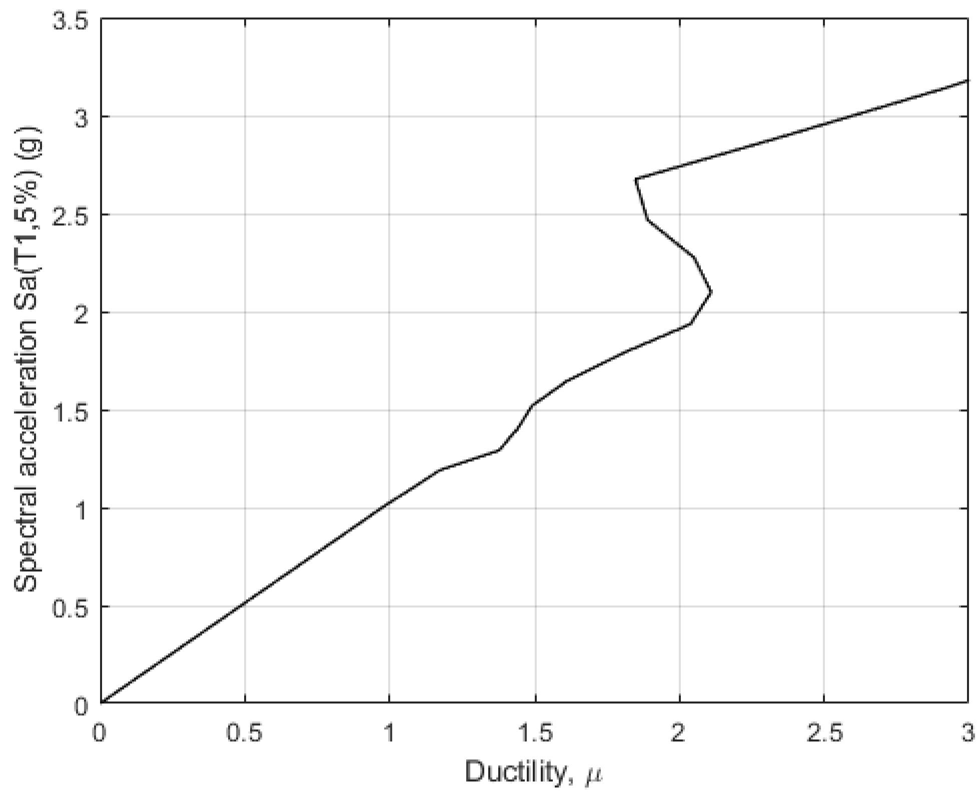
Apply OpenSeismoMatlab

```
S1=OpenSeismoMatlab(dt,xgtt,sw,T,lambdaF,IM_DM,m,uy,pysf,ksi,AlgID,...
    u0,ut0,rinf,maxtol,jmax,dak);
```

Plot the IDA curve

Initialize figure

```
figure()
% Plot the IDA curve
plot(S1.DM,S1.IM/9.81,'k','LineWidth',1)
% Finalize figure
grid on
xlabel('Ductility, \mu')
ylabel('Spectral acceleration Sa(T1,5%) (g)')
xlim([0,3])
ylim([0,3.5])
drawnow;
pause(0.1)
```



Copyright

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

Published with MATLAB® R2022b

verification Incremental dynamic analysis for ductility response

Contents

Reference

De Luca, F., Vamvatsikos, D., & Iervolino, I. (2011, May). Near-optimal bilinear fit of capacity curves for equivalent SDOF analysis. In Proceedings of the COMPDYN2011 Conference on Computational Methods in Structural Dynamics and Earthquake Engineering, Corfu, Greece.

Description

Figure 1(b) of the above reference presents the median IDA curves of SDOF systems with T=0.5sec. The actual capacity curve of the SDOF oscillator shown in Figure 1(a) of the same reference (green line), has been fitted with an elastoplastic bilinear fit according to FEMA-440 (blue line). This fitting introduces an error (bias) which appears as the blue area in Figure 1(b), which is generally conservative. In this example two arbitrary acceleration time histories are selected, then the corresponding displacement response IDA curves are plotted, based on a SDOF system with suitably selected properties, based on Figure 1(a). It is shown that both curves approximately fall into the bias (blue area) of Figure 1(b) of the above reference.

Earthquake motions

Load data from two earthquakes

```
GM={'Imperial Valley'; % Imperial valley 1979
    'Cape Mendocino'};
n=size(GM,1);
dt=cell(n,1);
xgtt=cell(n,1);
for i=1:n
    fid=fopen([GM{i},'.dat'],'r');
    text=textscan(fid,'%f %f');
    fclose(fid);
    t=text{1,1};
    dt{i}=t(2)-t(1);
    xgtt{i}=text{1,2};
end
```

Setup parameters for IDA analysis

Switch

```
sw='ida';
```

Eigenperiod

```
T=0.5;
```

Scaling factors

```
lambdaF=logspace(log10(0.01),log10(30),100);
```

Type of IDA analysis

```
IM_DM='Sa_disp';
```

Yield displacement

```
uy=0.042;
```

Initial stiffness

```
k_hi=1000/uy;
```

Mass

```
m=k_hi/(2*pi/T)^2;
```

Post yield stiffness factor

```
pysf=0.01;
```

Fraction of critical viscous damping

```
ksi=0.05;
```

Algorithm to be used for the time integration

```
AlgID='U0-V0-Opt';
```

Set initial displacement

```
u0=0;
```

Set initial velocity

```
ut0=0;
```

Minimum absolute value of the eigenvalues of the amplification matrix

```
rinf=1;
```

Maximum tolerance for convergence

```
maxtol=0.01;
```

Maximum number of iterations per increment

```
jmax=200;
```

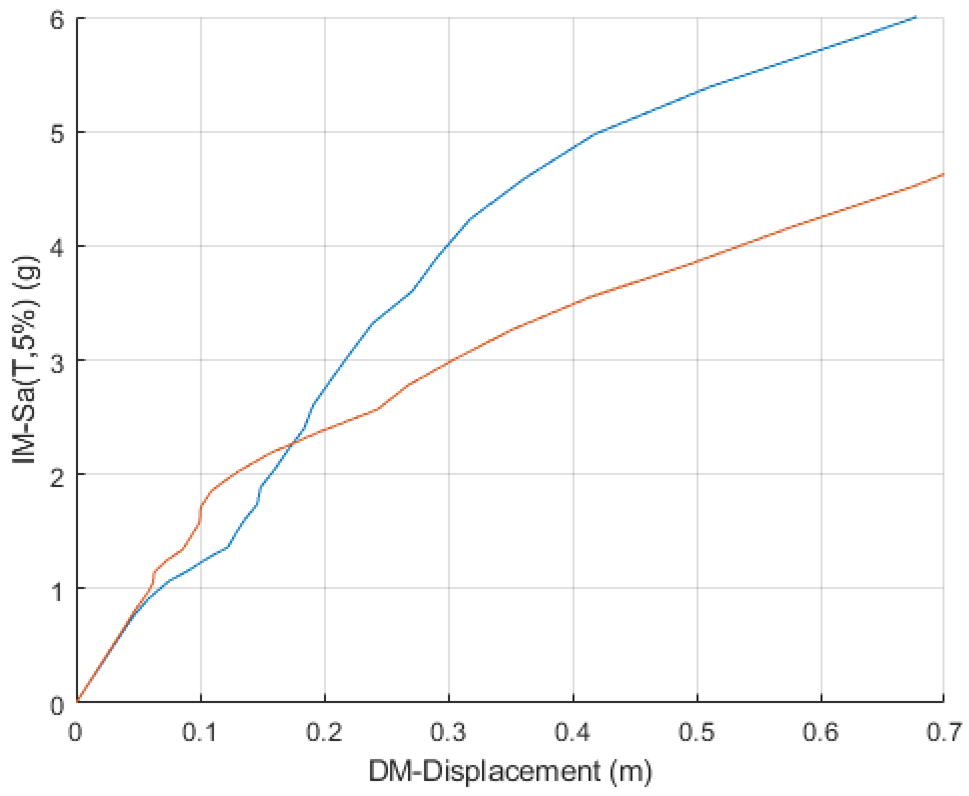
Infinitesimal variation of acceleration

```
dak=eps;
```

Construct and plot the IDA curves in a loop

Initialize figure

```
figure()
hold on
% Plot the IDA curves of Figure 1(b) of the above reference
for i=1:n
    S1=OpenSeismoMatlab(dt{i},xgtt{i},sw,T,lambdaF,IM_DM,m,uy,pysf,ksi,AlgID,...
        u0,ut0,rinf,maxtol,jmax,dak);
    plot(S1.DM,S1.IM/9.81)
end
% Finalize figure
grid on
xlabel('DM-Displacement (m)')
ylabel('IM-Sa(T,5%) (g)')
xlim([0,0.7])
ylim([0,6])
drawnow;
pause(0.1)
```



Copyright

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

Published with MATLAB® R2022b

verification Linear dynamic response of SDOF oscillator

Calculate the dynamic response of a linear SDOF oscillator. This example verifies Figure 6.6.1 in Chopra for Tn=1sec

Contents

Reference

Chopra, A. K. (2020). Dynamics of structures, Theory and Applications to Earthquake Engineering, 5th edition. Prenctice Hall.

Earthquake motion

Load earthquake data

```
dt=0.02;
fid=fopen('elcentro_NS_trunc.dat','r');
text=textscan(fid,'%f %f');
fclose(fid);
xgtt=text{1,2};
```

Setup parameters for LIDA function

Eigenperiod

```
Tn=1;
```

Critical damping ratio

```
ksi=0.02;
```

Initial displacement

```
u0=0;
```

Initial velocity

```
ut0=0;
```

Algorithm to be used for the time integration

```
AlgID='U0-V0-Opt';
```

Minimum absolute value of the eigenvalues of the amplification matrix

```
rinf=1;
```

Calculate dynamic response

Calculate circular eigenfrequency

```
omega=2*pi/Tn;
```

Apply LIDA

```
[u,ut,utt] = LIDA(dt,xgtt,omega,ksi,u0,ut0,AlgID,rinf);
```

Results

Maximum displacement in cm

```
D=max(abs(u))*100
```

```
D =

          15.0632751930384
```

Copyright

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

verification of seismic input energy of linear SDOF oscillator

Compare the output of LIDA.m and NLIDABLKIN.m for a linear SDOF oscillator in terms of seismic input energy per unit mass.

Contents

Reference

None

Description

Calculate the time histories of the various energies that are absorbed by a linear elastic SDOF oscillator using the linear function LIDA.m and the nonlinear function NLIDABLKIN.m to which an infinite yield limit has been assigned. The time history of the seismic input energy per unit mass obtained by the LIDA.m function (Figure b) is compared to the sum of the strain, damping and kinetic energies per unit mass obtained by the NLIDABLKIN.m function (Figure a). The linear SDOF system has Tn=0.5 sec and ksi=5%.

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

Setup parameters for NLIDABLKIN function for linear SDOF

Mass

```
m=1;
```

Eigenperiod

```
Tn=0.5;
```

Calculate the small-strain stiffness matrix

```
omega=2*pi/Tn;
k_hi=m*omega^2;
```

Assign linear elastic properties

```
k_lo=k_hi;
uy1=1e10;
```

Critical damping ratio

```
ksi=0.05;
```

Initial displacement

```
u0=0;
```

Initial velocity

```
ut0=0;
```

Algorithm to be used for the time integration

```
AlgID='U0-V0-Opt';
```

Minimum absolute value of the eigenvalues of the amplification matrix

```
rinf=1;
```

Maximum tolerance of convergence for time integration algorithm

```
maxtol=0.01;
```

Maximum number of iterations per integration time step

```
jmax=200;
```

Infinitesimal acceleration

```
dak=eps;
```

Calculate dynamic response of linear SDOF using NLIDABLKIN.m

Apply NLIDABLKIN

```
[u,ut,utt,Fs,Ey,Es,Ed,jiter] = NLIDABLKIN(dt,xgtt,m,k_hi,k_lo,uy1,...
    ksi,AlgID,u0,ut0,rinf,maxtol,jmax,dak);
```

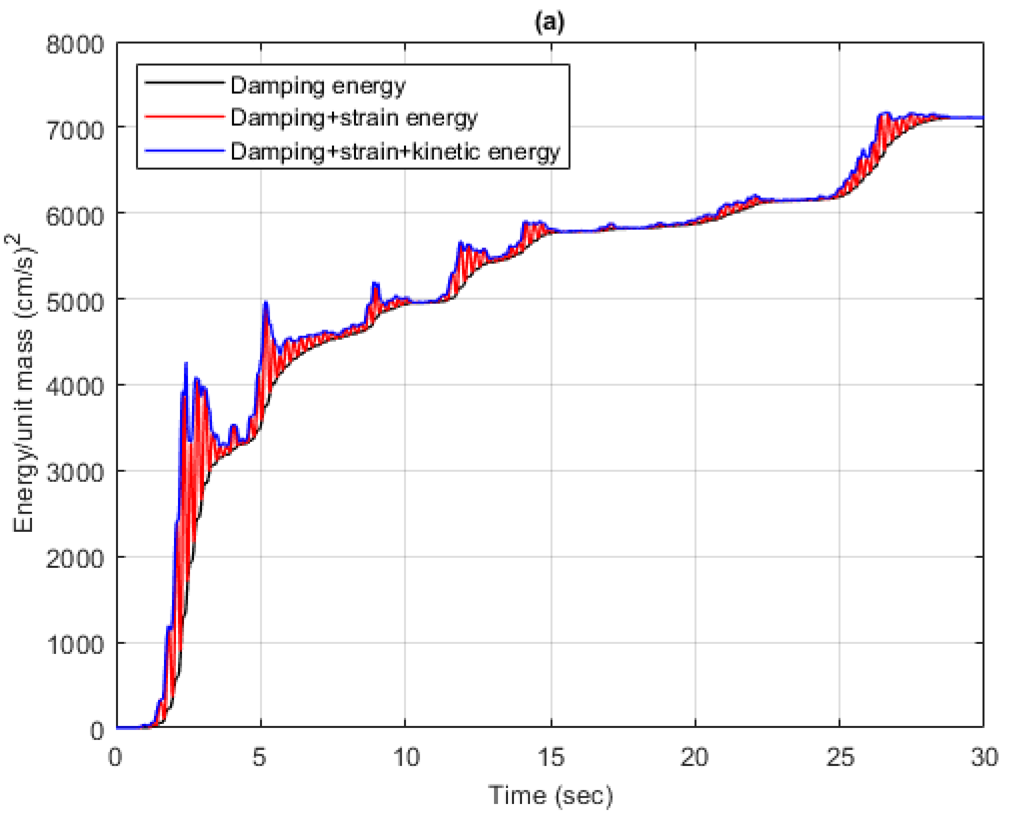
Calculate the kinetic energy of SDOF

```
Ek=1/2*m*ut.^2;
```

Plot the energy time histories of the linear SDOF

Plot the damping energy, damping plus strain energy, damping plus strain plus kinetic energy of the linearly elastic SDOF system. Convert from m to cm

```
figure()
plot(t',cumsum(Ed)*1e4,'k','LineWidth',1)
hold on
plot(t',cumsum(Ed)*1e4+Es*1e4,'r','LineWidth',1)
plot(t',cumsum(Ed)*1e4+Es*1e4+Ek*1e4,'b','LineWidth',1)
hold off
xlim([0,30])
ylim([0,8000])
xlabel('Time (sec)','FontSize',10);
ylabel('Energy/unit mass (cm/s)^2','FontSize',10);
title('(a)','FontSize',10)
grid on
legend({'Damping energy','Damping+strain energy',...
    'Damping+strain+kinetic energy'},'location','northwest')
drawnow;
pause(0.1)
```



Calculate dynamic response of the linear SDOF using LIDA.m

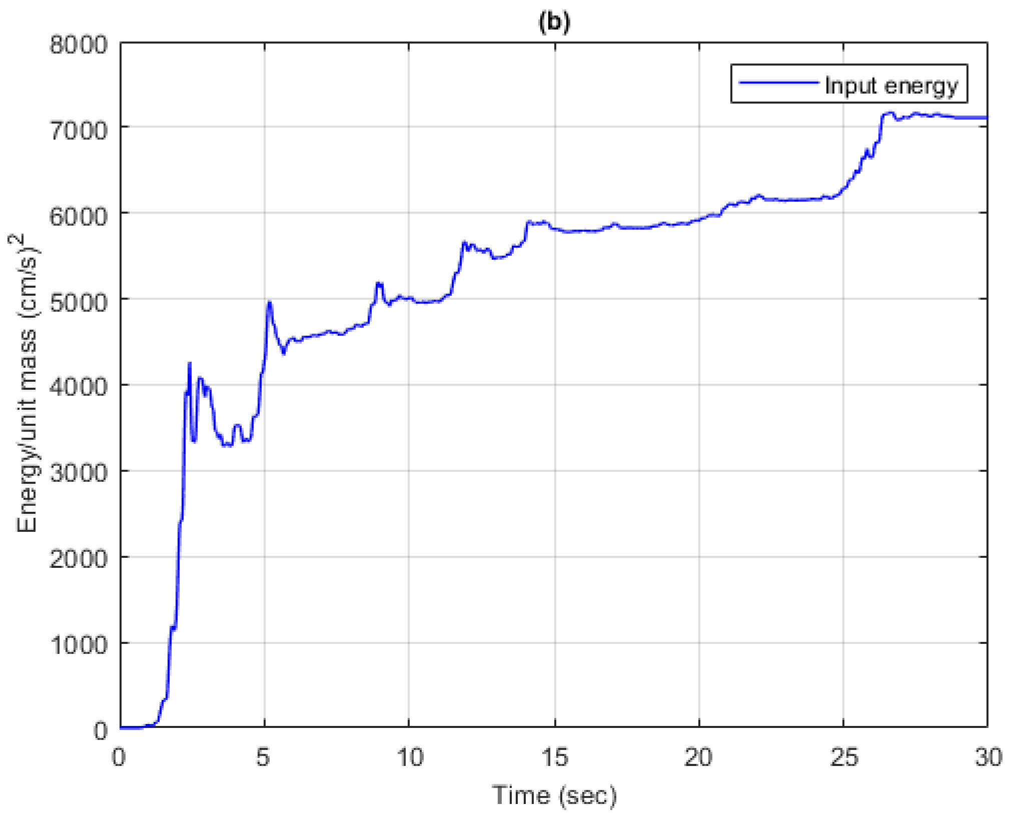
Apply the LIDA.m function for the linear SDOF

```
[u,ut,utt,Ei] = LIDA(dt,xgtt,omega,ksi,u0,ut0,AlgID,rinf);
```

Plot the energy time history of the nonlinear SDOF

Plot the damping energy, the hysteretic energy and strain energy of the nonlinear SDOF system. Convert from m to cm.

```
figure();
plot(t,cumsum(Ei)*1e4,'b','LineWidth',1)
xlim([0,30])
ylim([0,8000])
xlabel('Time (sec)','FontSize',10);
ylabel('Energy/unit mass (cm/s)^2','FontSize',10);
title('(b)','FontSize',10)
grid on
legend('Input energy')
drawnow;
pause(0.1)
```



Copyright

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

Published with MATLAB® R2022b

verification Energy time history of SDOF oscillator

Calculate the time history of the strain energy and the energy dissipated by viscous damping and yielding of a linear and a nonlinear SDOF oscillator

Contents

Reference

Chopra, A. K. (2020). Dynamics of structures, Theory and Applications to Earthquake Engineering, 5th edition. Prenctice Hall.

Description

Figure 7.9.1 of the above reference is reproduced in this example, for both the linear elastic and the elastoplastic SDOF systems. The linear system has Tn=0.5 sec and ksi=5%, whereas the elastoplastic system has Tn=0.5 sec, ksi=5% and fybar=0.25.

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

Setup parameters for NLIDABLKIN function for linear SDOF

Mass

```
m=1;
```

Eigenperiod

```
Tn=0.5;
```

Calculate the small-strain stiffness matrix

```
omega=2*pi/Tn;
k_hi=m*omega^2;
```

Assign linear elastic properties

```
k_lo=k_hi;
uy1=1e10;
```

Critical damping ratio

```
ksi=0.05;
```

Initial displacement

```
u0=0;
```

Initial velocity

```
ut0=0;
```

Algorithm to be used for the time integration

```
AlgID='U0-V0-Opt';
```

Minimum absolute value of the eigenvalues of the amplification matrix

```
rinf=1;
```

Maximum tolerance of convergence for time integration algorithm

```
maxtol=0.01;
```

Maximum number of iterations per integration time step

```
jmax=200;
```

Infinitesimal acceleration

```
dak=eps;
```

Calculate dynamic response of the linear SDOF

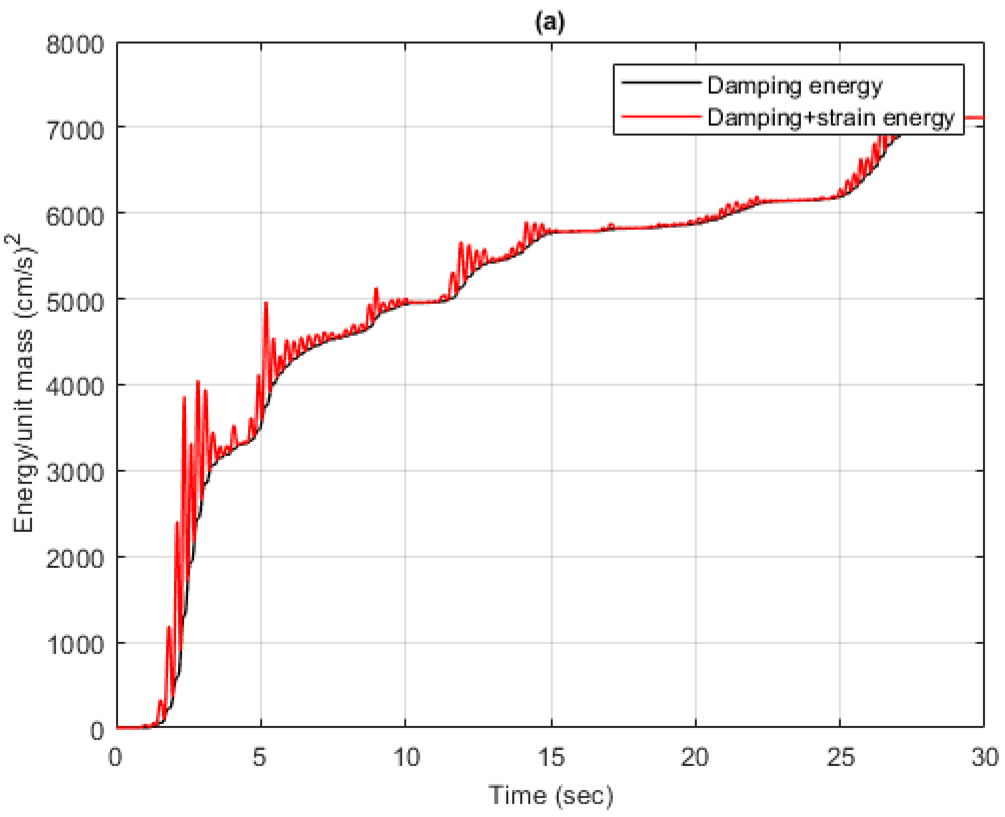
Apply NLIDABLKIN

```
[u,ut,utt,Fs,Ey,Es,Ed,jiter] = NLIDABLKIN(dt,xgtt,m,k_hi,k_lo,uy1,...
    ksi,AlgID,u0,ut0,rinf,maxtol,jmax,dak);
```

Plot the energy time history of the linear SDOF

Plot the damping energy and strain energy of the linearly elastic SDOF system. Convert from m to cm

```
figure()
plot(t',cumsum(Ed)*1e4,'k','LineWidth',1)
hold on
plot(t',cumsum(Ed)*1e4+Es*1e4,'r','LineWidth',1)
hold off
xlim([0,30])
ylim([0,8000])
xlabel('Time (sec)','FontSize',10);
ylabel('Energy/unit mass (cm/s)^2','FontSize',10);
title('(a)','FontSize',10)
grid on
legend('Damping energy','Damping+strain energy')
drawnow;
pause(0.1)
```



Setup parameters for NLIDABLKIN function for nonlinear SDOF

The properties of the nonlinear SDOF system are identical to those of the linear SDOF system, except for the yield displacement and the post-yield stiffness.

Post yield stiffness

```
k_lo=0.01*k_hi;
```

normalized yield strength

```
fybar=0.25;
```

Yield displacement for nonlinear response

```
uy2=fybar*max(abs(u));
```

Calculate dynamic response of the nonlinear SDOF

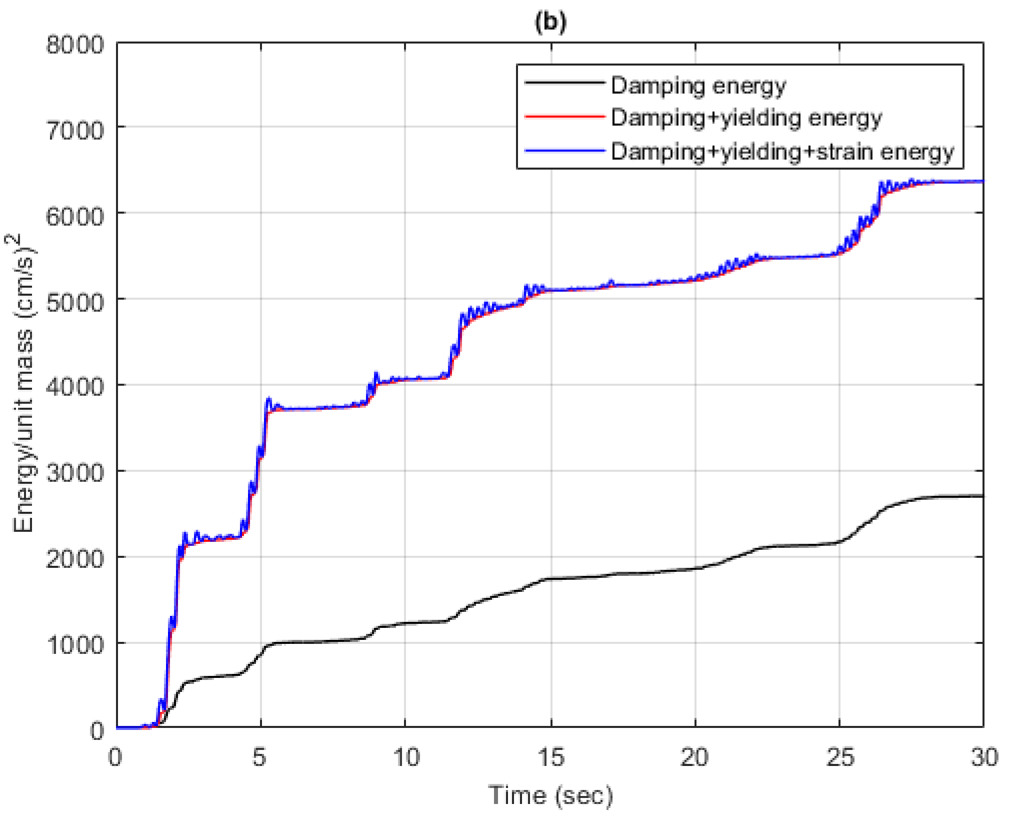
Apply NLIDABLKIN

```
[u,ut,utt,Fs,Ey,Es,Ed,jiter] = NLIDABLKIN(dt,xgtt,m,k_hi,k_lo,uy2,...
    ksi,AlgID,u0,ut0,rinf,maxtol,jmax,dak);
```

Plot the energy time history of the nonlinear SDOF

Plot the damping energy, the hysteretic energy and strain energy of the nonlinear SDOF system. Convert from m to cm.

```
figure();
plot(t',cumsum(Ed)*1e4,'k','LineWidth',1)
hold on
plot(t',cumsum(Ed)*1e4+cumsum(Ey)*1e4-Es*1e4,'r','LineWidth',1)
plot(t',cumsum(Ed)*1e4+cumsum(Ey)*1e4,'b','LineWidth',1)
hold off
xlim([0,30])
ylim([0,8000])
xlabel('Time (sec)','FontSize',10);
ylabel('Energy/unit mass (cm/s)^2','FontSize',10);
title('(b)','FontSize',10)
grid on
legend('Damping energy','Damping+yielding energy',...
    'Damping+yielding+strain energy')
drawnow;
pause(0.1)
```



Copyright

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

verification Energy time histories of nonlinear SDOF oscillator

Calculate the time histories of the energy dissipated by damping, the energy dissipated by yielding, as well as the energy that is input to two SDOF oscillators with two different eigenperiods.

Contents

Reference

Uang, C. M., & Bertero, V. V. (1990). Evaluation of seismic energy in structures. Earthquake engineering & structural dynamics, 19(1), 77-90.

Description

Figure 3(b) of the above reference is reproduced in this example, for both linear elastic-perfectly plastic SDOF systems (short period and long period). Thetwo eigenperiods considered are T=0.2 sec and T=5.0 sec. The damping ratio is 5% of the critical value and the ductility ratio achieved is equal to 5 for both SDOF systems. Figure 3(b) is verified in terms of damping energy (noted as E_ksi in the Figure), yielding energy (noted as E_h in the Figure) and input energy (noted as Ei in the Figure).

Load earthquake data

Earthquake acceleration time history of the 1986 San Salvador earthquake will be used (San Salvador, 10/10/1986, Geotech Investigation Center, component 90)

```
fid=fopen('SanSalvador1986GIC090.txt','r');
text=textscan(fid,'%f %f');
fclose(fid);
t=text{1,1};
dt=t(2)-t(1);
xgtt=text{1,2};
```

Setup parameters for NLIDABLKIN function for nonlinear SDOF with T=0.2s

Mass

```
m=1;
```

Eigenperiod

```
T=0.2;
```

Calculate the small-strain stiffness matrix

```
omega=2*pi/T;
k_hi=m*omega^2;
```

Assign post-yield stiffness

```
k_lo=0.0001*k_hi;
```

Assign yield limit

```
uy=0.004875;
```

Critical damping ratio

```
ksi=0.05;
```

Initial displacement

```
u0=0;
```

Initial velocity

```
ut0=0;
```

Algorithm to be used for the time integration

```
AlgID='U0-V0-Opt';
```

Minimum absolute value of the eigenvalues of the amplification matrix

```
rinf=1;
```

Maximum tolerance of convergence for time integration algorithm

```
maxtol=0.01;
```

Maximum number of iterations per integration time step

```
jmax=200;
```

Infinitesimal acceleration

```
dak=eps;
```

Calculate dynamic response of the nonlinear SDOF with T=0.2s

Apply NLIDABLKIN

```
[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);
```

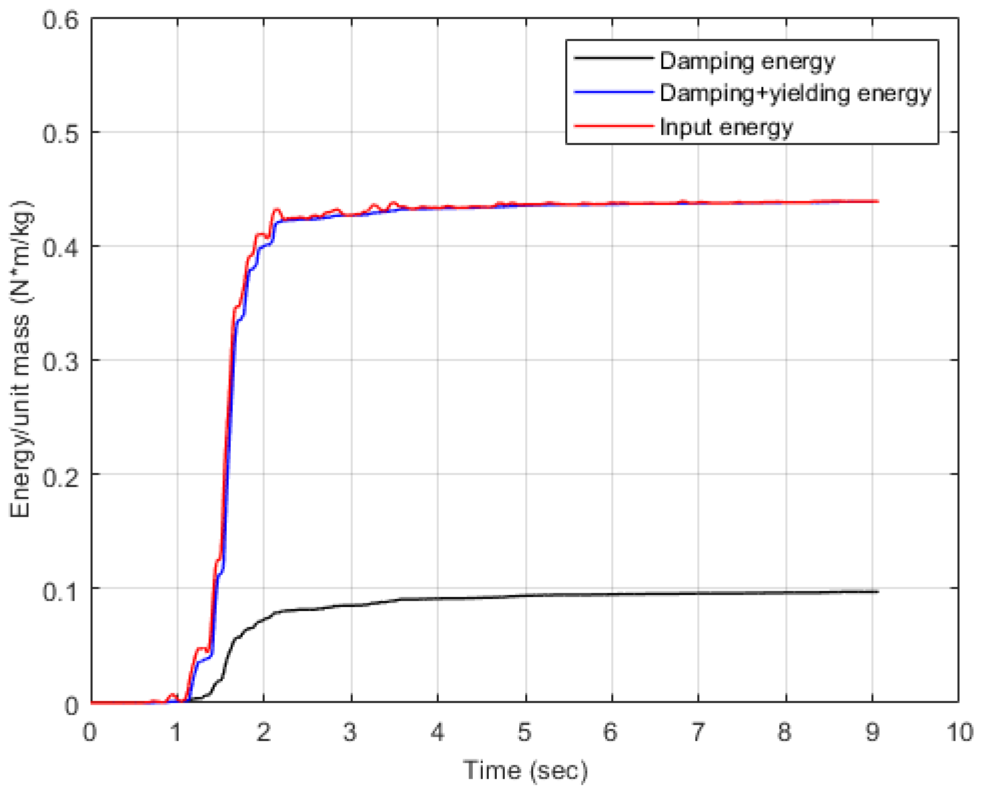
Calculate the kinetic energy of SDOF

```
Ek=1/2*m*ut.^2;
```

Plot the energy time histories of the nonlinear SDOF with T=0.2s

Plot the damping energy, yielding energy and input energy

```
figure()
plot(t',cumsum(Ed),'k','LineWidth',1)
hold on
plot(t',cumsum(Ed)+cumsum(Ey)-Es,'b','LineWidth',1)
plot(t',cumsum(Ed)+cumsum(Ey)+Ek,'r','LineWidth',1)
hold off
xlim([0,10])
ylim([0,0.6])
xlabel('Time (sec)','FontSize',10);
ylabel('Energy/unit mass (N*m/kg)','FontSize',10);
%title('(a)','FontSize',10)
grid on
legend('Damping energy','Damping+yielding energy','Input energy')
drawnow;
pause(0.1)
```



Show the achieved ductility factor (must be equal to 5 according to the above reference)

```
max(abs(u))/uy
```

```
ans =

          5.00074829936011
```

Setup parameters for NLIDABLKIN function for nonlinear SDOF with T=0.5s

Mass

```
m=1;
```

Eigenperiod

```
T=5;
```

Calculate the small-strain stiffness matrix

```
omega=2*pi/T;
k_hi=m*omega^2;
```

Assign post-yield stiffness

```
k_lo=0.0001*k_hi;
```

Assign yield limit

```
uy=0.023;
```

Critical damping ratio

```
ksi=0.05;
```

Initial displacement

```
u0=0;
```

Initial velocity

```
ut0=0;
```

Algorithm to be used for the time integration

```
AlgID='U0-V0-Opt';
```

Minimum absolute value of the eigenvalues of the amplification matrix

```
rinf=1;
```

Maximum tolerance of convergence for time integration algorithm

```
maxtol=0.01;
```

Maximum number of iterations per integration time step

```
jmax=200;
```

Infinitesimal acceleration

```
dak=eps;
```

Calculate dynamic response of the nonlinear SDOF with T=0.5s

Apply NLIDABLKIN

```
[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);
```

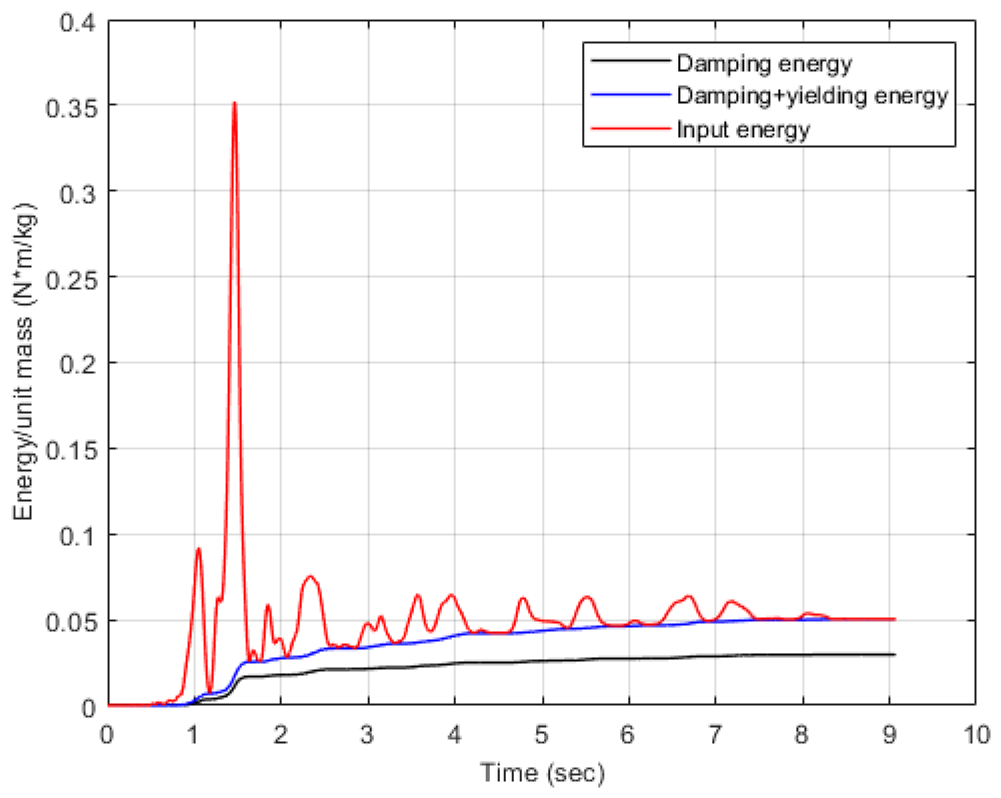
Calculate the kinetic energy of SDOF

```
Ek=1/2*m*ut.^2;
```

Plot the energy time histories of the nonlinear SDOF with T=0.5s

Plot the damping energy, yielding energy and input energy

```
figure()
plot(t',cumsum(Ed),'k','LineWidth',1)
hold on
plot(t',cumsum(Ed)+cumsum(Ey)-Es,'b','LineWidth',1)
plot(t',cumsum(Ed)+cumsum(Ey)+Ek,'r','LineWidth',1)
hold off
xlim([0,10])
%ylim([0,0.6])
xlabel('Time (sec)','FontSize',10);
ylabel('Energy/unit mass (N*m/kg)','FontSize',10);
%title('(a)','FontSize',10)
grid on
legend('Damping energy','Damping+yielding energy','Input energy')
drawnow;
pause(0.1)
```



Show the achieved ductility factor (must be equal to 5 according to the above reference)

```
max(abs(u))/uy
```

```
ans =

         4.99190405960856
```

Copyright

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

verification Strength ductility relation of elastoplastic system

Contents

Reference

Mahin, S. A., & Lin, J. (1983). Construction of inelastic response spectra for single-degree-of-freedom systems. Earthquake Engineering Center, University of California, Berkeley.

Description

Figure 15 of the above reference is reproduced in this example, for an elastic perfectly plastic (EPP) system with eigenperiod equal to 0.7 sec and damping ratio equal to 0.05. The NS component of the El Centro earthquake (1940) is considered. It is demonstrated that there can be more than one values of strength corresponding to a given displacement ductility.

Load earthquake data

Earthquake acceleration time history of the El Centro earthquake will be used (El Centro, 1940, NS component)

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

Setup parameters for NLIDABLKIN function for linear SDOF

Mass

```
m=1;
```

Eigenperiod

```
Tn=0.7;
```

Calculate the small-strain stiffness matrix

```
omega=2*pi/Tn;
k_hi=m*omega^2;
```

Assign linear elastic properties

```
k_lo=0.001*k_hi;
```

Assign yield displacement

```
eta=[0.3,0.5,0.8,0.9,1.0,1.3,1.5];
uy=eta*max(abs(xgtt))/k_hi;
```

Critical damping ratio

```
ksi=0.05;
```

Initial displacement

```
u0=0;
```

Initial velocity

```
ut0=0;
```

Algorithm to be used for the time integration

```
AlgID='U0-V0-Opt';
```

Minimum absolute value of the eigenvalues of the amplification matrix

```
rinf=1;
```

Maximum tolerance of convergence for time integration algorithm

```
maxtol=0.01;
```

Maximum number of iterations per integration time step

```
jmax=200;
```

Infinitesimal acceleration

```
dak=eps;
```

Calculate dynamic response of the linear SDOF

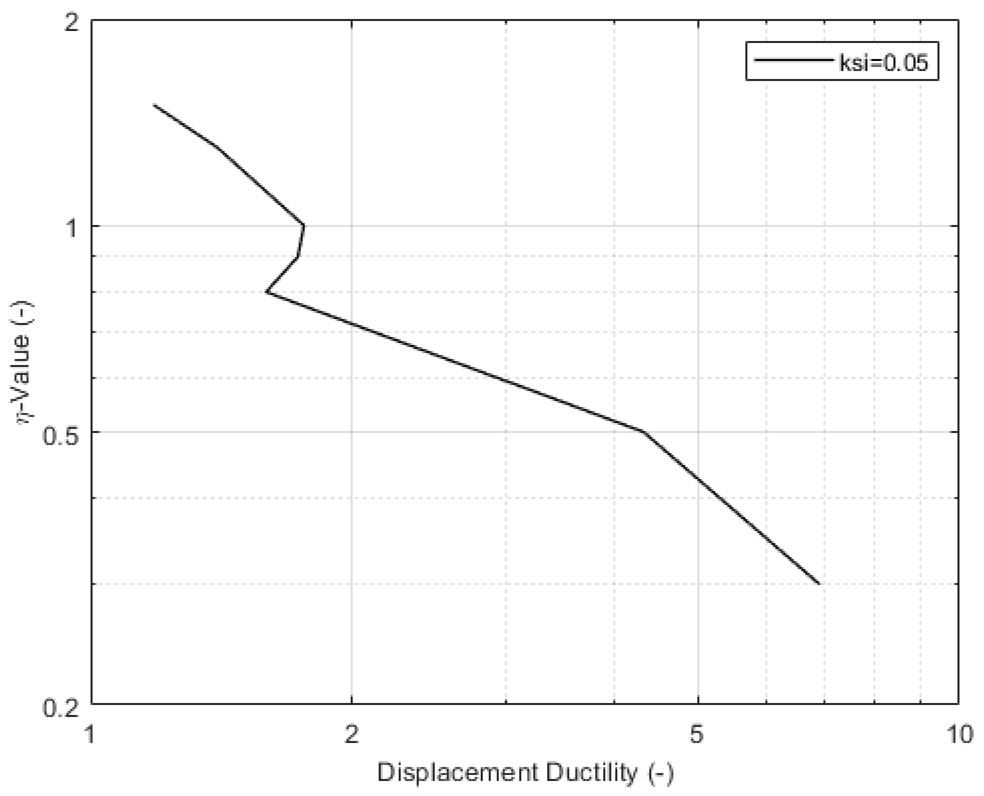
Apply NLIDABLKIN

```
mu=zeros(1,numel(uy));
for i=1:numel(uy)
[u,ut,utt,Fs,Ey,Es,Ed,jiter] = NLIDABLKIN(dt,xgtt,m,k_hi,k_lo,uy(i),...
    ksi,AlgID,u0,ut0,rinf,maxtol,jmax,dak);
    mu(i)=max(abs(u))/uy(i);
end
```

Plot the energy time history of the linear SDOF

Plot the damping energy and strain energy of the linearly elastic SDOF system. Convert from m to cm

```
figure()
loglog(mu,eta,'k','LineWidth',1)
xlim([1,10])
ylim([0.2,2])
set(gca, 'XTick', [1,2,5,10])
set(gca, 'YTick', [0.2,0.5,1,2])
xlabel('Displacement Ductility (-)','FontSize',10);
ylabel('\eta-Value (-)','FontSize',10);
grid on
legend({'ksi=0.05'})
drawnow;
pause(0.1)
```



Copyright

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

verification Pulse Decomposition of OpenSeismoMatlab

Contents

Reference

Baker, J. W. (2007). Quantitative classification of near-fault ground motions using wavelet analysis. Bulletin of the seismological society of America, 97(5), 1486-1501.

Description

The velocity pulse of a strong motion velocity time history is extracted according to Figure 4 of the above reference. The velocity time history of the fault-normal component of the 1994 Northridge, Rinaldi, recording is considered. The Daubechies wavelet of order 4, shown in Figure 2c, is used as the mother wavelet. Initially, the largest velocity pulse is identified (Figure 4b). For the ground motion under consideration, the ten largest velocity pulses are extracted, the combination of which is shown in Figure 4c, and the residual ground motion after the pulses have been removed is shown (Figure 4d).

Earthquake motion

Load earthquake data of fault-normal component of the 1994 Northridge, Rinaldi, recording

```
eqmotion={'RSN1063_NORTHR_RRS228.txt'};
data=load(eqmotion{1});
t=data(:,1);
dt=t(2)-t(1);
xgt=data(:,2);
```

Perform modification so that the initial velocity time history agrees with the plot of Figure 4a of the above reference.

```
xgt=-xgt;
```

Perform pulse decomposition of earthquake motion

Switch

```
sw='pulsedecomp';
```

Wavelet family of Daubechies wavelet of order 4

```
wname = 'db4';
```

Apply OpenSeismoMatlab for 1st largest pulse decomposition

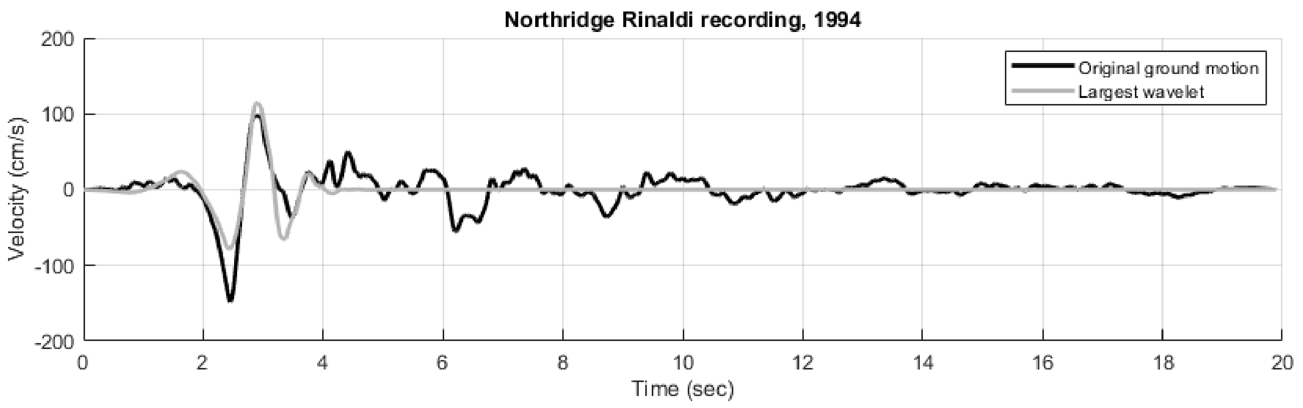
```
S1=OpenSeismoMatlab(dt,xgt,sw,wname);
```

Plot the velocity time histories (initial motion and largest pulse)

Initialize figure

```
figure('Position',[7.4,508.2,1517.6,253.8],...
    'InnerPosition',[7.4,508.2,1517.6,253.8],...
    'OuterPosition',[0.2,501,1000,343.2])
hold on
```

```
% Plot the velocity time history of the initial ground motion
plot(t,xgt, 'Color', [0 0 0], 'LineWidth', 2)
% Plot the time history of the largest velocity pulse
plot(t,S1.pulseTH, 'Color', [0.7 0.7 0.7], 'LineWidth', 2)
% Finalize figure
hold off
grid on
title('Northridge Rinaldi recording, 1994')
ylim([-200,200])
xlim([0,20])
xlabel('Time (sec)')
ylabel('Velocity (cm/s)')
legend({'Original ground motion','Largest wavelet'})
drawnow;
pause(0.1)
```



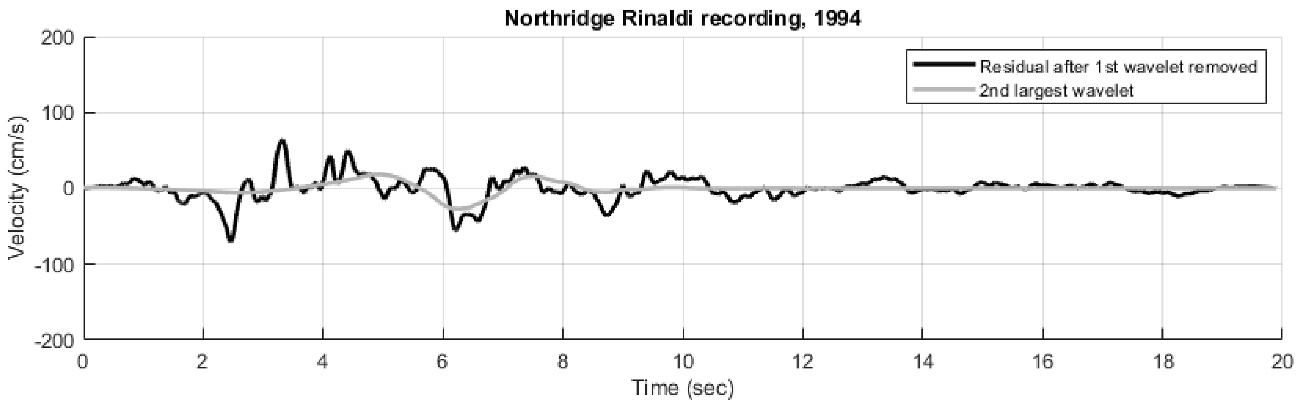
Apply OpenSeismoMatlab for extracting the 2nd up to 10th largest pulses from the residual motion after extracting the 1st pulse as above

```
Sout{1}=S1;
for i=2:10
    S1=OpenSeismoMatlab(dt,S1.resTH,sw,wname);
    Sout{i}=S1;
end
```

Plot the velocity time histories (residual motion and 2nd largest pulse)

Initialize figure

```
figure('Position',[7.4,508.2,1517.6,253.8],...
    'InnerPosition',[7.4,508.2,1517.6,253.8],...
    'OuterPosition',[0.2,501,1000,343.2])
hold on
% Plot the velocity time history of the residual ground motion
plot(t,Sout{1}.resTH, 'Color', [0 0 0], 'LineWidth', 2)
% Plot the time history of the largest velocity pulse
plot(t,Sout{2}.pulseTH, 'Color', [0.7 0.7 0.7], 'LineWidth', 2)
% Finalize figure
hold off
grid on
title('Northridge Rinaldi recording, 1994')
ylim([-200,200])
xlim([0,20])
xlabel('Time (sec)')
ylabel('Velocity (cm/s)')
legend({'Residual after 1st wavelet removed','2nd largest wavelet'})
drawnow;
pause(0.1)
```



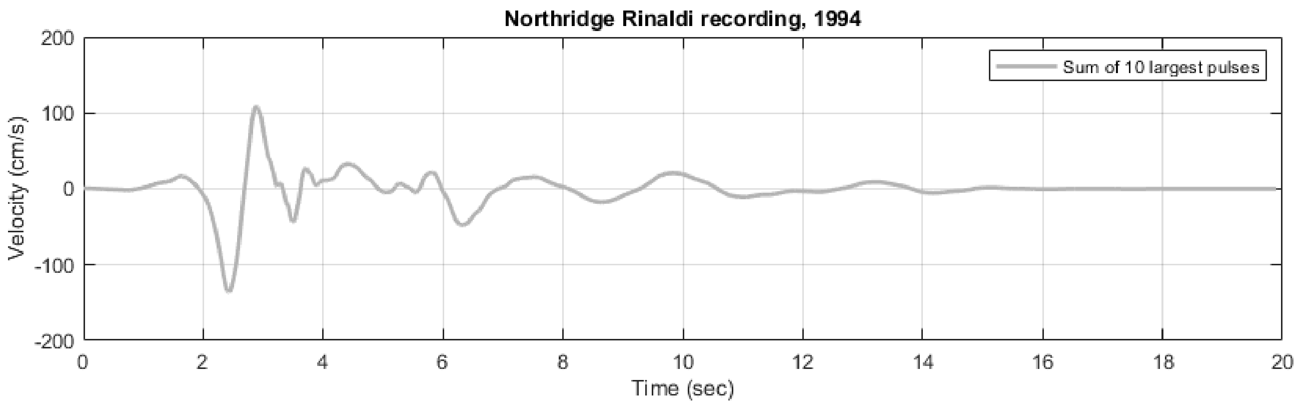
Sum the 10 largest pulses extracted from the input motion

```
sum10=zeros(size(S1.pulseTH));
for i=1:10
    sum10=sum10+Sout{i}.pulseTH;
end
```

Plot the velocity pulse using 10 largest pulses

Initialize figure

```
figure('Position',[7.4,508.2,1517.6,253.8],...
    'InnerPosition',[7.4,508.2,1517.6,253.8],...
    'OuterPosition',[0.2,501,1000,343.2])
% Plot the time history of the largest velocity pulse
plot(t,sum10, 'Color', [0.7 0.7 0.7], 'LineWidth', 2)
% Finalize figure
grid on
title('Northridge Rinaldi recording, 1994')
ylim([-200,200])
xlim([0,20])
xlabel('Time (sec)')
ylabel('Velocity (cm/s)')
legend({'Sum of 10 largest pulses'})
drawnow;
pause(0.1)
```

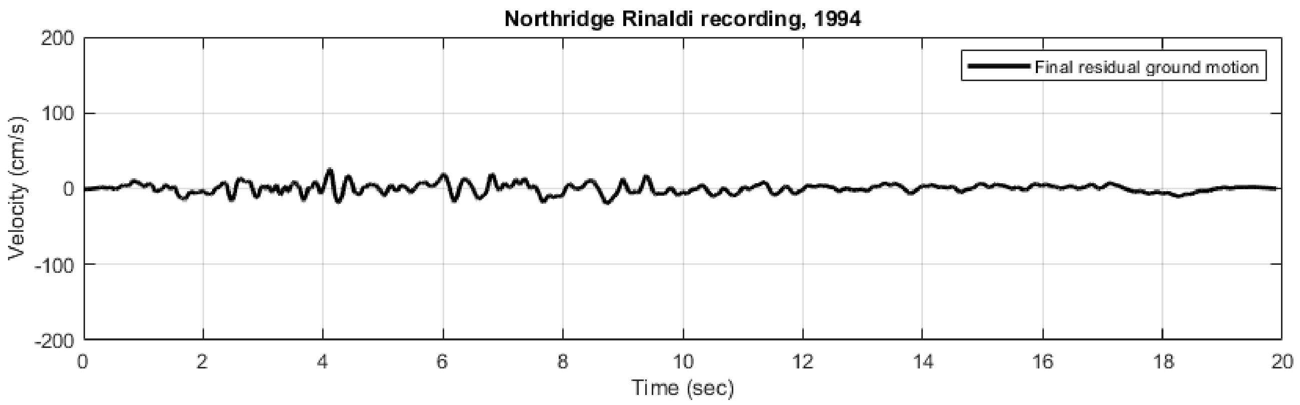


Plot the final residual ground motion

Initialize figure

```
figure('Position',[7.4,508.2,1517.6,253.8],...
    'InnerPosition',[7.4,508.2,1517.6,253.8],...
    'OuterPosition',[0.2,501,1000,343.2])
% Plot the time history of the largest velocity pulse
plot(t,S1.resTH, 'Color', [0 0 0], 'LineWidth', 2)
% Finalize figure
grid on
title('Northridge Rinaldi recording, 1994')
```

```
ylim([-200,200])
xlim([0,20])
xlabel('Time (sec)')
ylabel('Velocity (cm/s)')
legend({'Final residual ground motion'})
drawnow;
pause(0.1)
```



Copyright

verification Pulse Decomposition of OpenSeismoMatlab

Contents

Reference

Shahi, S. K., & Baker, J. W. (2014). An efficient algorithm to identify strong-velocity pulses in multicomponent ground motions. Bulletin of the Seismological Society of America, 104(5), 2456-2466.

Description

The velocity pulses of a strong motion velocity time history are extracted according to Figure 3 of the above reference, and the extracted pulses are summed to give the reconstructed ground motion. There is significant agreement between the original and the reconstructed ground motions. The velocity time history of the 1979 Imperial Valley El Centro Array 4 recording is considered. The Daubechies wavelet of order 4 is used as the mother wavelet.

Earthquake motion

Load earthquake data of 1979 Imperial Valley El Centro Array 4 recording

```
eqmotion={'ImperialValleyElCentroArray4.txt'};
data=load(eqmotion{1});
t=data(:,1);
dt=t(2)-t(1);
xgt=data(:,2);
```

Perform pulse decomposition of earthquake motion

Switch

```
sw='pulsedecomp';
```

Wavelet family of Daubechies wavelet of order 4

```
wname = 'db4';
```

Apply OpenSeismoMatlab for extracting the first 30 largest pulses from the initial motion

```
S1=OpenSeismoMatlab(dt,xgt,sw,wname);
Sout{1}=S1;
for i=2:30
    S1=OpenSeismoMatlab(dt,S1.resTH,sw,wname);
    Sout{i}=S1;
end
```

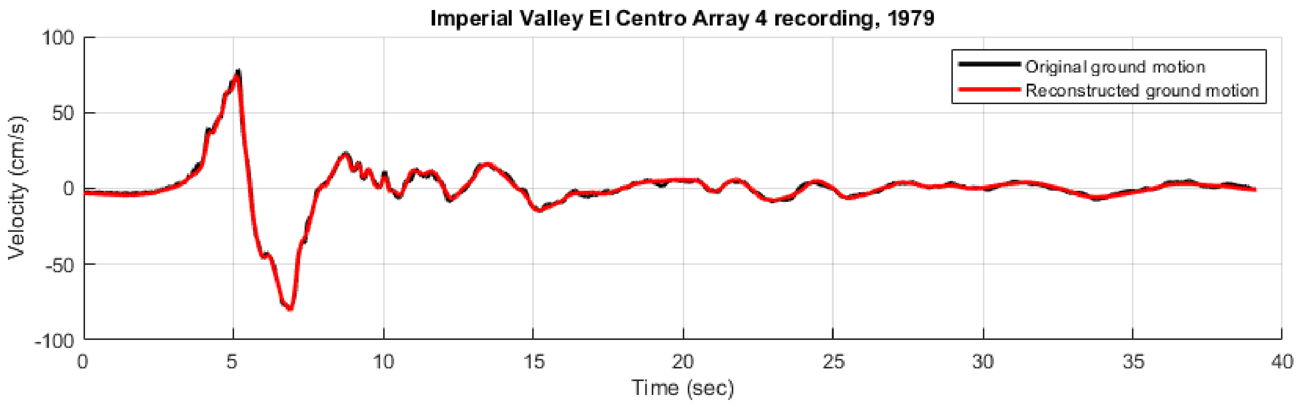
Sum the 30 largest pulses extracted from the input motion

```
sum30=zeros(size(S1.pulseTH));
for i=1:30
    sum30=sum30+Sout{i}.pulseTH;
end
```

Plot the velocity time histories (original and reconstructed)

Initialize figure

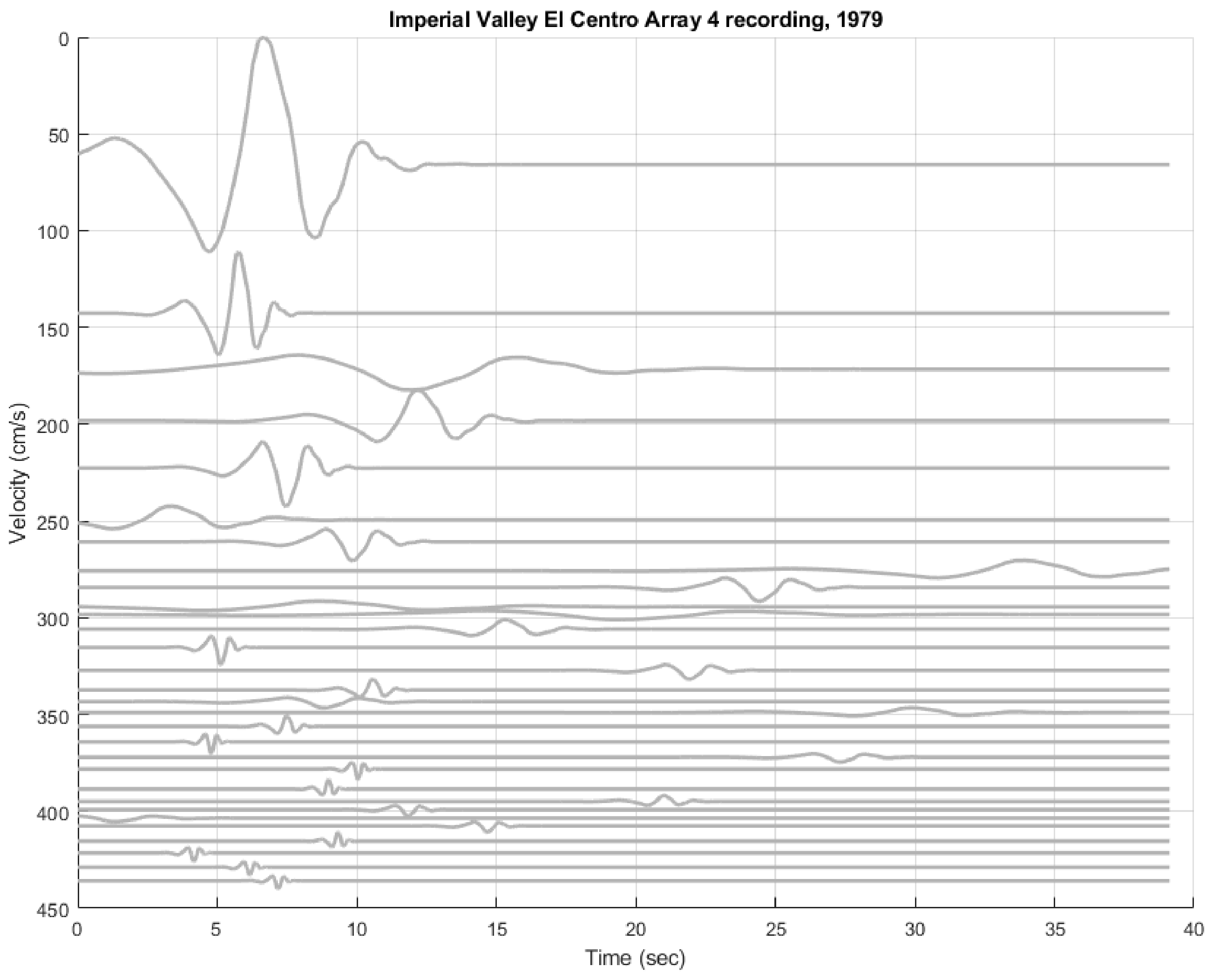
```
figure('Position',[7.4,508.2,1517.6,253.8],...
    'InnerPosition',[7.4,508.2,1517.6,253.8],...
    'OuterPosition',[0.2,501,1000,343.2])
hold on
% Plot the velocity time history of the original ground motion
plot(t,xgt, 'Color', [0 0 0], 'LineWidth', 2)
% Plot the velocity time history of the reconstructed ground motion
plot(t,sum30, 'Color', [1 0 0], 'LineWidth', 2)
% Finalize figure
hold off
grid on
title('Imperial Valley El Centro Array 4 recording, 1979')
ylim([-100,100])
xlim([0,40])
xlabel('Time (sec)')
ylabel('Velocity (cm/s)')
legend({'Original ground motion','Reconstructed ground motion'})
drawnow;
pause(0.1)
```



Plot the 30 velocity pulses in a single plot

Verify with the middle plot of Figure 3 Initialize figure

```
figure('Position',[5     50.6    1524.8   731.2],...
    'InnerPosition',[5  50.6    1524.8   731.2],...
    'OuterPosition',[-2.20      43.4    1000    820.8])
hold on
offSet=0;
for i=1:30
    offSet=offSet+abs(min(Sout{i}.pulseTH));
    plot(t,offSet+Sout{i}.pulseTH, 'Color', [0.7 0.7 0.7], 'LineWidth', 2)
    offSet=offSet+max(Sout{i}.pulseTH);
end
% Finalize figure
hold off
grid on
title('Imperial Valley El Centro Array 4 recording, 1979')
ylim([0,450])
set(gca, 'YDir','reverse')
xlim([0,40])
xlabel('Time (sec)')
ylabel('Velocity (cm/s)')
drawnow;
pause(0.1)
```



Copyright

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

Published with MATLAB® R2022b

verification Spectral Intensity according to Housner (1952)

Calculate the spectral intensity as defined by Housner (1952) using OpenSeismoMatlab

Contents

Reference

Housner, G. W. (1952). Intensity of ground motion during strong earthquakes. California Institute of Technology, Second technical report, Office of naval research, task order 25, project NR-081-095.

Description

Table II on page 15 of the above reference is verified for the case of the El Centro, 1940 earthquake.The spectral intensity according to this table is 8.35ft for damping 0.0, 2.71ft for damping 0.2 and 1,89ft for damping 0.4. These values are verified with the present example

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

Calculate Housner Spectral Intensity

Switch

```
sw='SIH1952';
```

First value of damping (Table II in the above reference)

```
ksi1=0.0;
```

Second value of damping (Table II in the above reference)

```
ksi2=0.2;
```

Third value of damping (Table II in the above reference)

```
ksi3=0.4;
```

Apply OpenSeismoMatlab for calculating of the spectral intensity

```
S1=OpenSeismoMatlab(dt,xgtt,sw,ksi1);
S2=OpenSeismoMatlab(dt,xgtt,sw,ksi2);
S3=OpenSeismoMatlab(dt,xgtt,sw,ksi3);
```

Spectral intensities in ft

For damping 0.0

```
S1.SI*3.28084
```

```
ans =

          8.00634608331064
```

For damping 0.2

```
S2.SI*3.28084
```

```
ans =

          2.33622318056444
```

For damping 0.4

```
S3.SI*3.28084
```

```
ans =

          1.63462066423459
```

Copyright

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

verification Rigid plastic sliding response spectrum

Contents

Reference

Garini, E., & Gazetas, G. (2016). Rocking and Sliding Potential of the 2014 Cephalonia, Greece Earthquakes. In ICONHIC2016 1st international conference on natural hazards & infrastructure June, Chania, Greece.

Description

The rigid plastic sliding response spectra are extracted for various acceleration time histories of the 3 Feb 2014 13.34 EST Cephalonia earthquake. Three records are used: Chavriata (EW), Lixouri (EW) and Lixouri (NS). The oscillator is considered to be ideally rigid-plastic sliding on horizontal plane, as shown in Figure 4(a) of the above reference. The acceleration time histories that are used for the extraction of the spectra are plotted in this example and also shown in Figure 2 of the above reference [Chavriata (3 February) EW, Lixouri (3 February) EW, Lixouri (3 February) NS]. The rigid plastic sliding response spectra extracted in this example are compared to the corresponding spectra that appear in Figure 6 of the above reference. Note that The two horizontal and the vertical component of the various records are imposed in the above reference, whereas in OpenSeismoMatlab only the horizontal component is considered. Therefore there may be some differences between the results of the above reference and the results of OpenSeismoMatlab.

Earthquake motions

Load earthquake data of Chavriata EW record

```
eqmotion={'CHV1-20140203E'};
data=load([eqmotion{1},'.txt']);
t1=data(:,1);
dt1=t1(2)-t1(1);
xgtt1=data(:,2)/100;
% Truncate the record
ind=t1>23 & t1<33;
xgtt1=xgtt1(ind);
t1=t1(ind);
```

Load earthquake data of Lixouri EW record

```
eqmotion={'LXR1-20140203E'};
data=load([eqmotion{1},'.txt']);
t2=data(:,1);
dt2=t2(2)-t2(1);
xgtt2=data(:,2)/100;
% Truncate the record
ind=t2>23 & t2<33;
xgtt2=xgtt2(ind);
t2=t2(ind);
```

Load earthquake data of Lixouri NS record

```
eqmotion={'LXR1-20140203N'};
data=load([eqmotion{1},'.txt']);
t3=data(:,1);
dt3=t3(2)-t3(1);
xgtt3=data(:,2)/100;
% Truncate the record
ind=t3>23 & t3<33;
xgtt3=xgtt3(ind);
t3=t3(ind);
```

Calculate rigid plastic sliding response spectrum of earthquake motion

Switch

```
sw='rpsrs';
```

Coulomb friction coefficients

```
CF=[0.05;0.1;0.2;0.4;0.5];
```

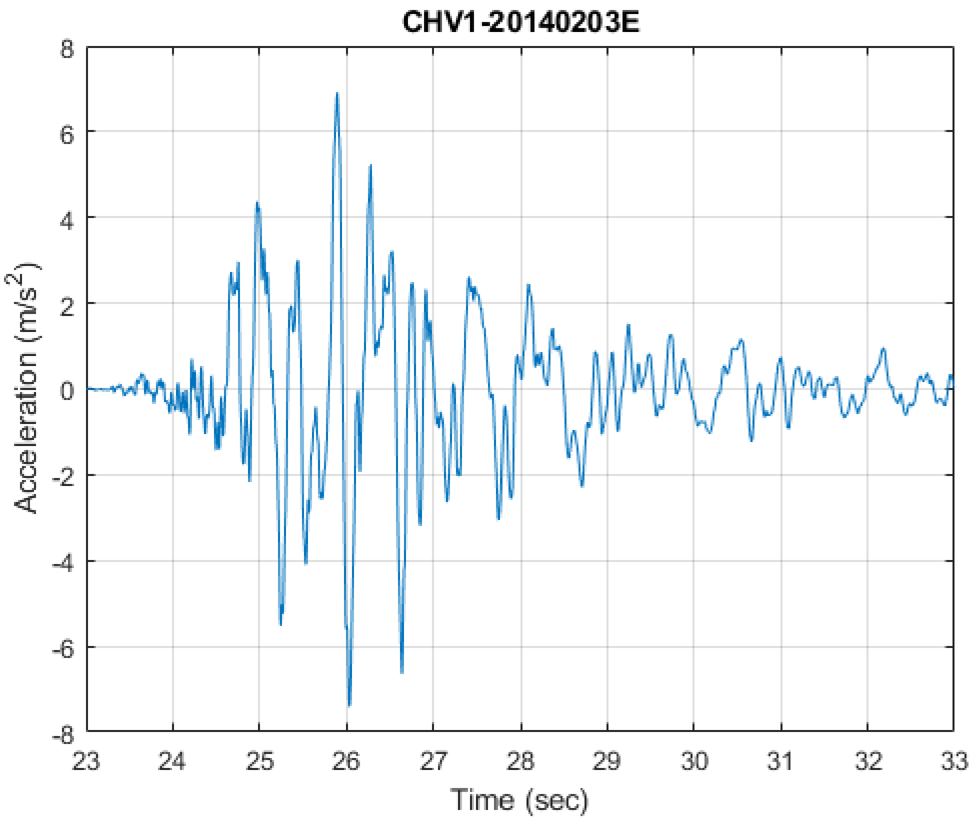
Apply OpenSeismoMatlab once for each record

```
S1=OpenSeismoMatlab(dt1,xgtt1,sw,CF);
S2=OpenSeismoMatlab(dt2,xgtt2,sw,CF);
S3=OpenSeismoMatlab(dt3,xgtt3,sw,CF);
```

Plot the acceleration time histories of the earthquake motions

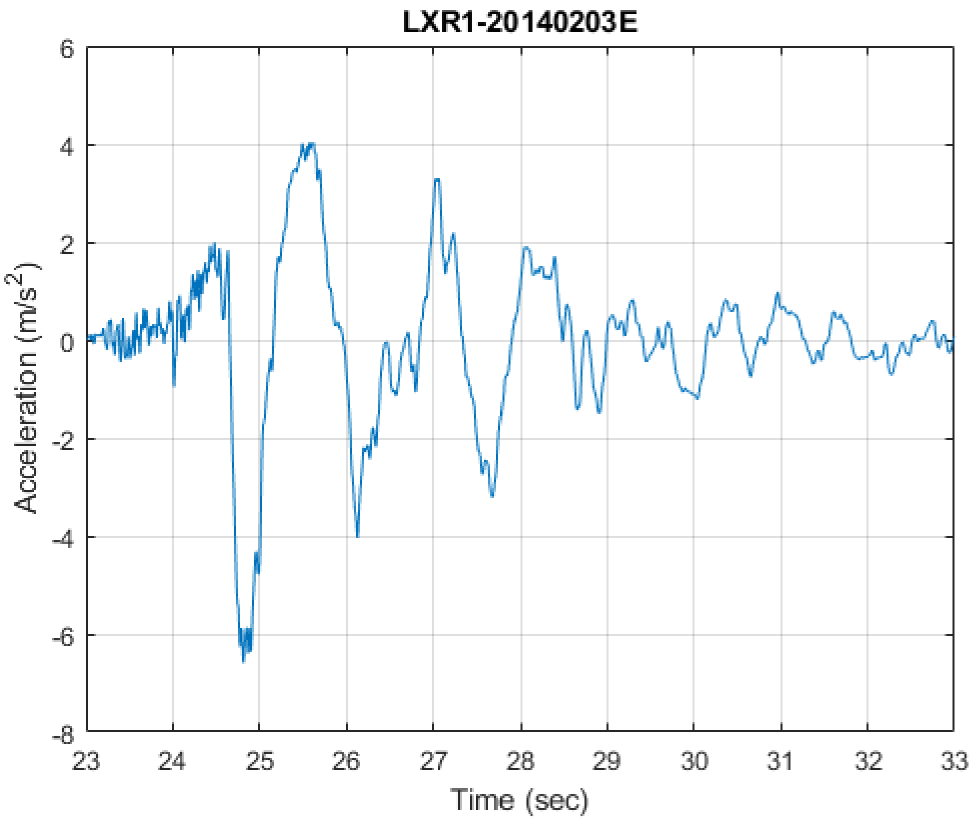
Initialize figure

```
figure()
% Plot the acceleration time history
plot(t1,xgtt1)
% Finalize figure
grid on
title('CHV1-20140203E')
xlabel('Time (sec)')
ylabel('Acceleration (m/s^2)')
drawnow;
pause(0.1)
```



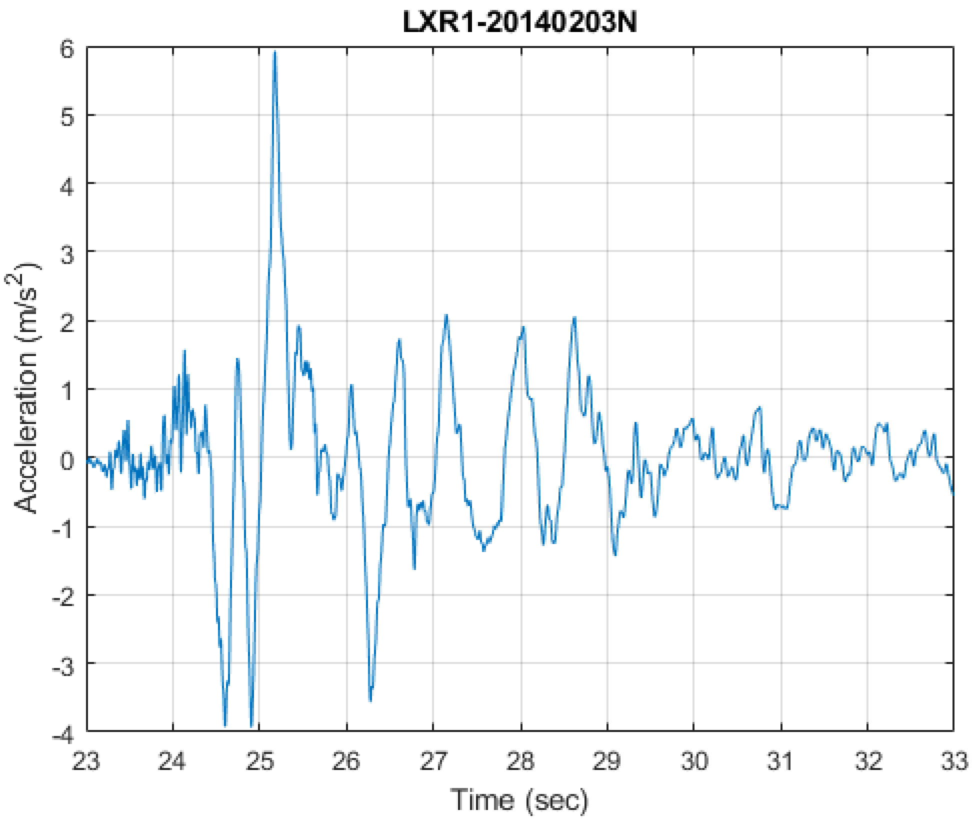
Initialize figure

```
figure()
% Plot the acceleration time history
plot(t2,xgtt2)
% Finalize figure
grid on
title('LXR1-20140203E')
xlabel('Time (sec)')
ylabel('Acceleration (m/s^2)')
drawnow;
pause(0.1)
```



Initialize figure

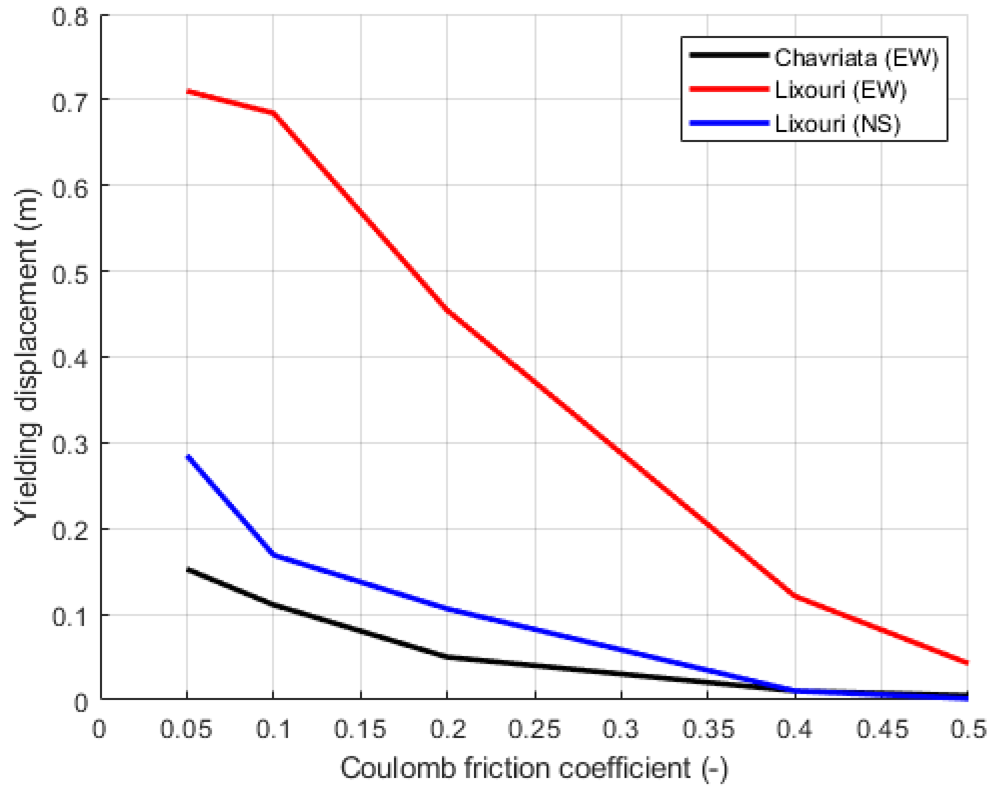
```
figure()
% Plot the acceleration time history
plot(t3,xgtt3)
% Finalize figure
grid on
title('LXR1-20140203N')
xlabel('Time (sec)')
ylabel('Acceleration (m/s^2)')
drawnow;
pause(0.1)
```



Plot the rigid plastic sliding response spectra

Initialize figure

```
figure()
hold on
% Plot the rigid plastic sliding response spectra
plot(S1.CF,S1.RPSSd, 'k-', 'LineWidth', 2)
plot(S2.CF,S2.RPSSd, 'r-', 'LineWidth', 2)
plot(S3.CF,S3.RPSSd, 'b-', 'LineWidth', 2)
hold off
% Finalize figure
grid on
xlabel('Coulomb friction coefficient (-)')
ylabel('Yielding displacement (m)')
legend({'Chavriata (EW)','Lixouri (EW)','Lixouri (NS)'})
xlim([0,0.5])
drawnow;
pause(0.1)
```



Chavriata (EW)
Lixouri (EW)
Lixouri (NS)
Yielding displacement (m)
Coulomb friction coefficient (-)
0
0.1
0.2
0.3
0.4
0.5
0.6
0.7
0.8
0
0.05
0.1
0.15
0.2
0.25
0.3
0.35
0.4
0.45
0.5

Copyright

verification Rigid plastic sliding response spectrum

Contents

Reference

Paglietti, A., & Porcu, M. C. (2001). Rigid–plastic approximation to predict plastic motion under strong earthquakes. Earthquake engineering & structural dynamics, 30(1), 115-126.

Description

The rigid plastic sliding response spectra are extracted for two acceleration time histories: the Northridge (Sylmar County), Jan 1994 record and the El Centro, NS, May 1940 record, as presented in Table 1 of the above reference. The oscillator is considered to be ideally rigid-plastic sliding on horizontal plane. The acceleration time histories that are used for the extraction of the spectra are plotted in this example. The rigid plastic sliding response spectra extracted in this example are compared to the corresponding spectra that appear in Figure 6 of the above reference.

Earthquake motions

Load earthquake data of Northridge Sylmar County 1994 record

```
eqmotion={'Northridge_Sylmar_County'};
data=load([eqmotion{1},'.dat']);
t1=data(:,1);
dt1=t1(2)-t1(1);
xgtt1=data(:,2);
```

Load earthquake data of El Centro 1940 record

```
eqmotion={'elcentro_NS_trunc'};
data=load([eqmotion{1},'.dat']);
t2=data(:,1);
dt2=t2(2)-t2(1);
xgtt2=data(:,2);
```

Calculate rigid plastic sliding response spectrum of earthquake motion

Switch

```
sw='rpsrs';
```

Coulomb friction coefficients

```
CF1=[0.006;0.05;0.1;0.15;0.3;0.4;0.7;0.8];
CF2=[0.002;0.01;0.05;0.1;0.15;0.3;0.4];
```

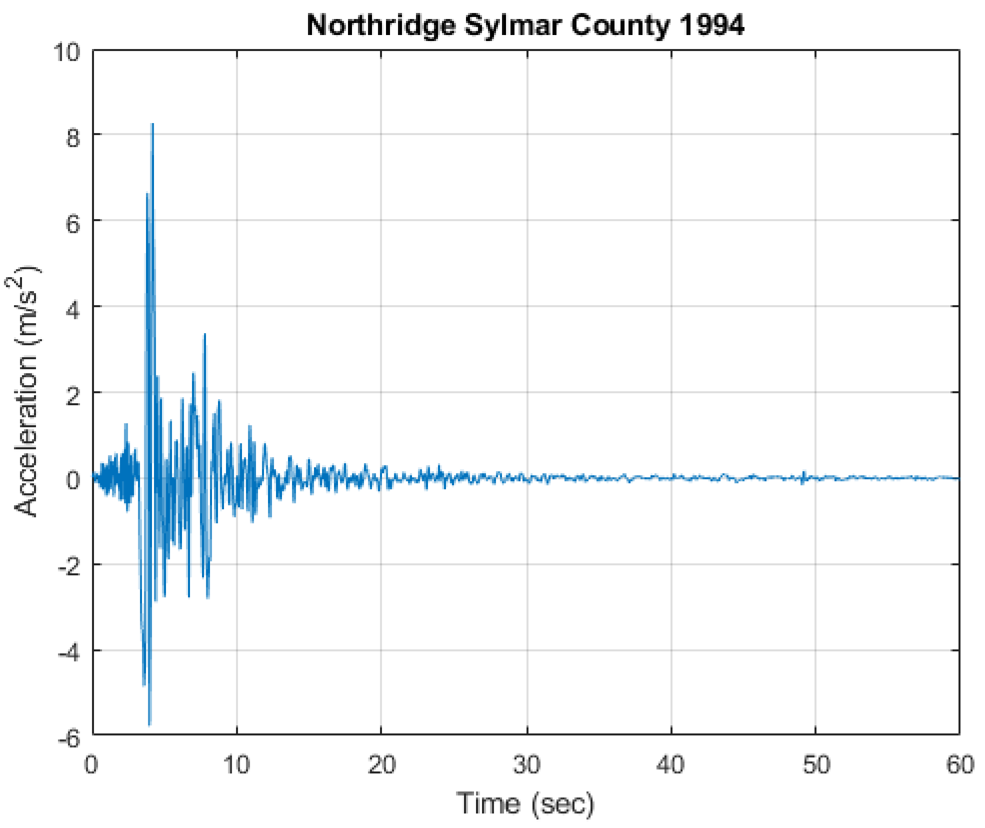
Apply OpenSeismoMatlab once for each record

```
S1=OpenSeismoMatlab(dt1,xgtt1,sw,CF1);
S2=OpenSeismoMatlab(dt2,xgtt2,sw,CF2);
```

Plot the acceleration time histories of the earthquake motions

Initialize figure

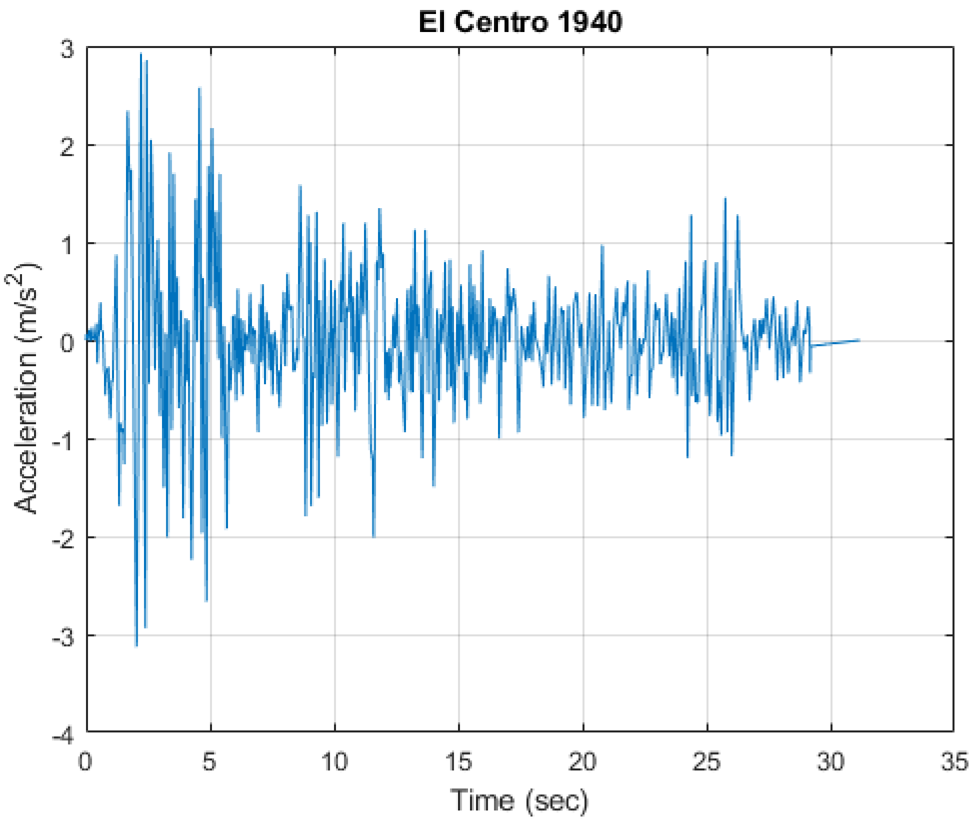
```
figure()
% Plot the acceleration time history
plot(t1,xgtt1)
% Finalize figure
grid on
title('Northridge Sylmar County 1994')
xlabel('Time (sec)')
ylabel('Acceleration (m/s^2)')
drawnow;
pause(0.1)
```



Initialize figure

```
figure()
% Plot the acceleration time history
```

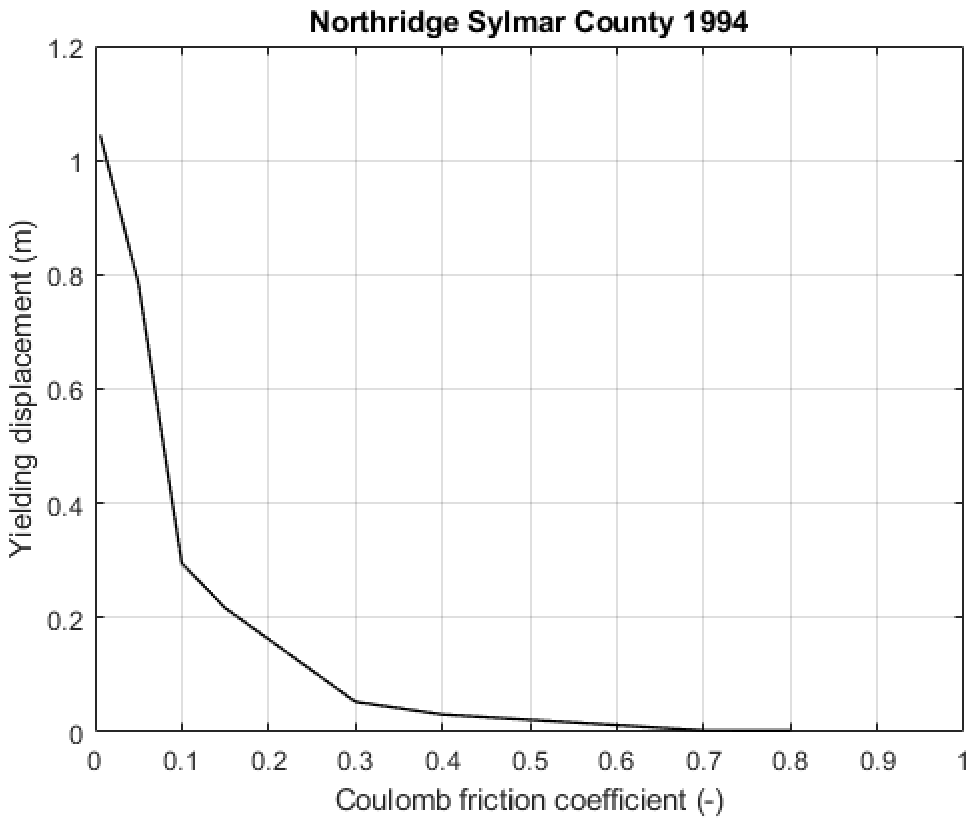
```
plot(t2,xgtt2)
% Finalize figure
grid on
title('El Centro 1940')
xlabel('Time (sec)')
ylabel('Acceleration (m/s^2)')
drawnow;
pause(0.1)
```



Plot the rigid plastic sliding response spectra

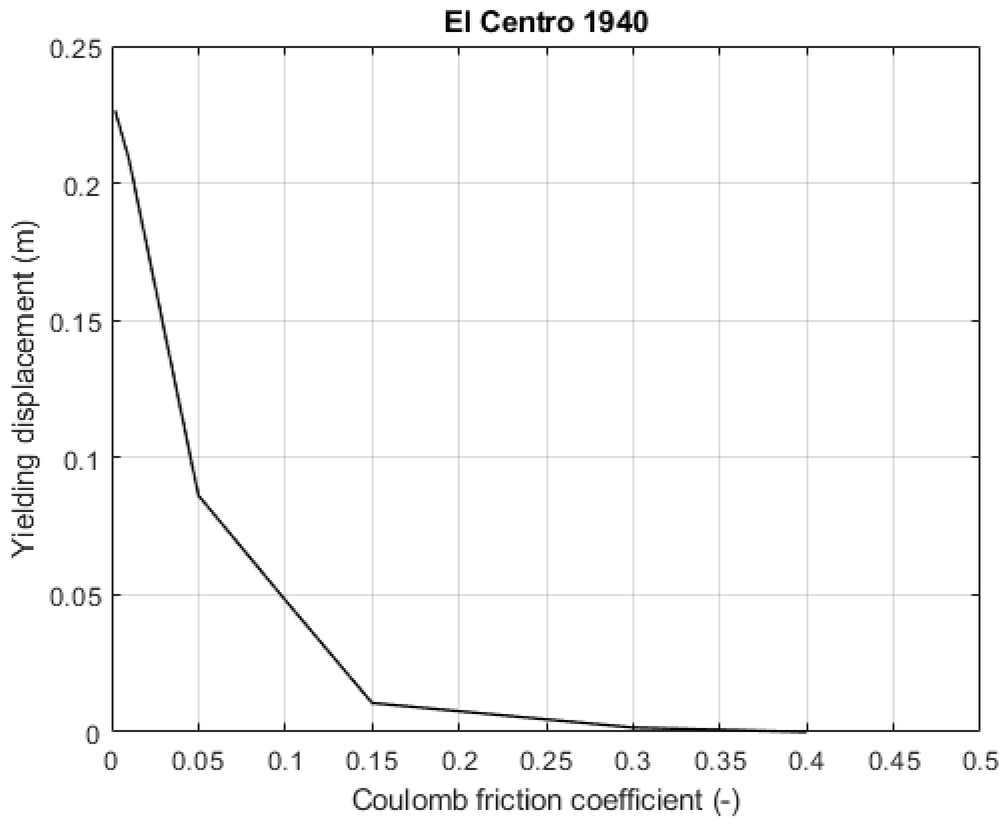
Initialize figure

```
figure()
% Plot the rigid plastic sliding response spectra
plot(S1.CF,S1.RPSSd, 'k-', 'LineWidth', 1)
% Finalize figure
grid on
xlabel('Coulomb friction coefficient (-)')
ylabel('Yielding displacement (m)')
title('Northridge Sylmar County 1994')
xlim([0,1])
drawnow;
pause(0.1)
```



Initialize figure

```
figure()
% Plot the rigid plastic sliding response spectra
plot(S2.CF,S2.RPSSd, 'k-', 'LineWidth', 1)
% Finalize figure
grid on
xlabel('Coulomb friction coefficient (-)')
ylabel('Yielding displacement (m)')
title('El Centro 1940')
xlim([0,0.5])
drawnow;
pause(0.1)
```



Copyright

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

Published with MATLAB® R2022b