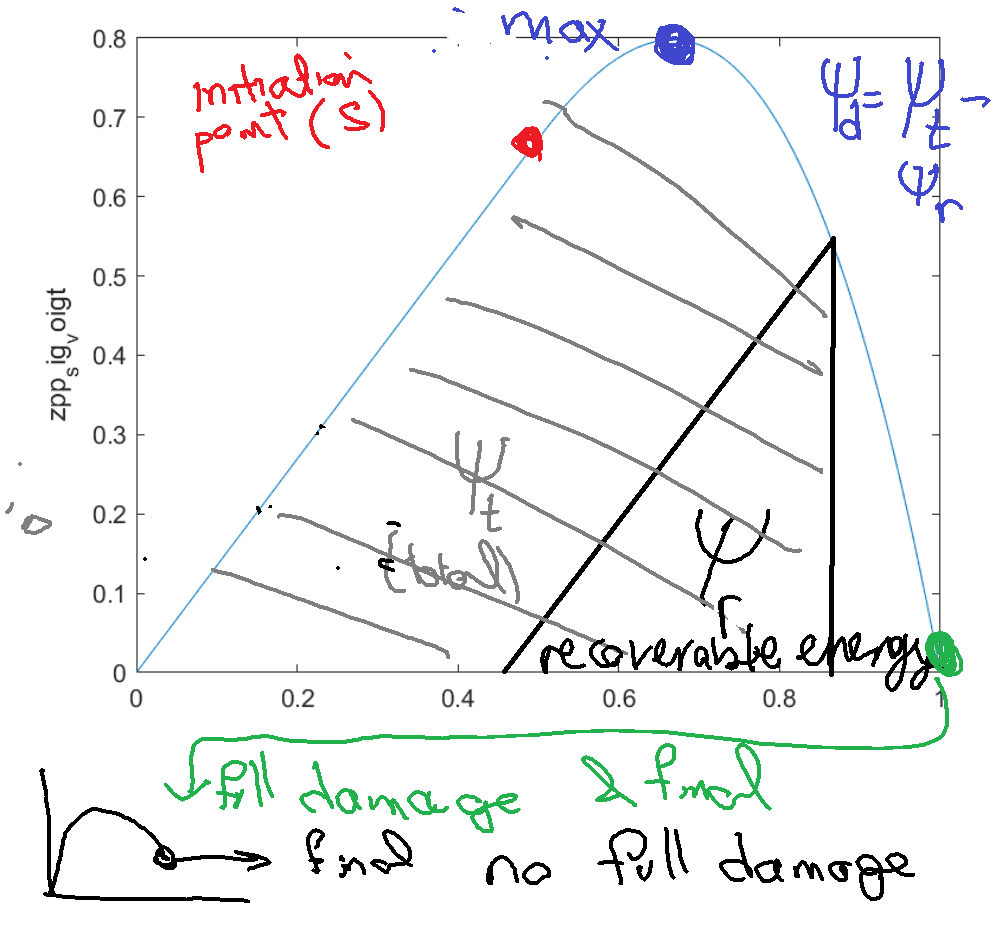
# Goal:

The class zpp\_DamageHomogenization1R finds critical points on a strain-stress curve, integrates certain quantities over time and strain, and homogenizes a bulk damage model from strain stress response.



The critical points and energy quantities are shown in the figure.

# 1. Sample run

Please run scriptTest\_zpr.m

Different steps are described below.

# 2. Input data:

1. raw strain stress history
2. Stiffness matrix (need to know when strain stress response deviations from linear response)
3. Any damage indicators in raw data set (e.g. crack length in aSDG runs; plasticity and damage measures in OSU data).
4. Any quantities that are to be integrated over data set time (or time step). For example, in dynamic problems the power at different time steps can be used to integrate and obtain energy.

Sets 3 and 4 can be empty. Set 4 is used for other initiation stage indicators (for example, what is the strain level when the first point in the domain experiences damage, or plasticity related to OSU results).

## How input data is stored:

Data history, that is strain, stress, damage indicators (if any), integrands, e.g. powers (if any) are stored in this class

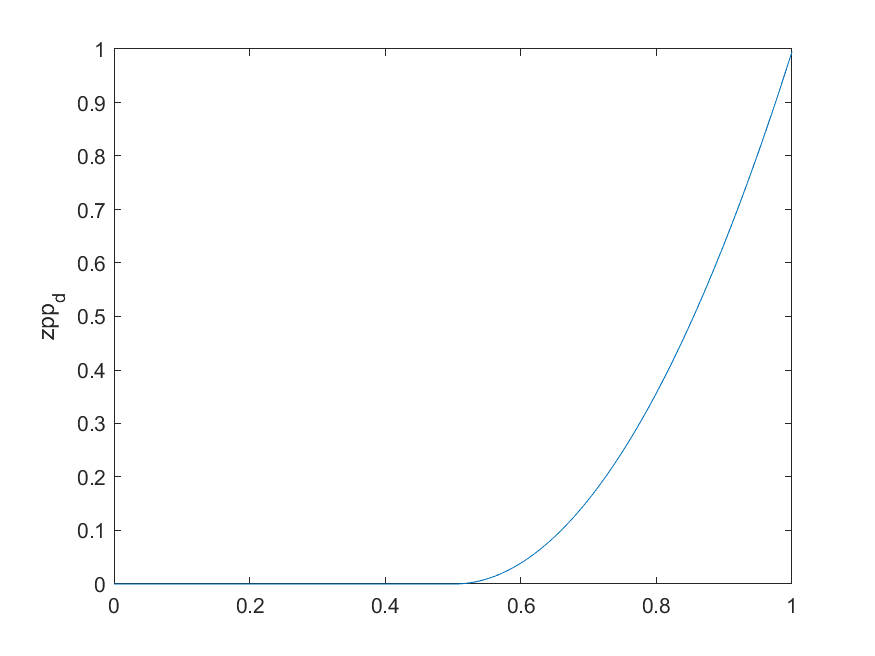
gen\_textIndexedDatasets

this class is basically a set of indexed data. Data is accessed by a given field name. For example, say give me stress at serial number (time step) 35. So, we directly work with field names, rather than giving them an index.

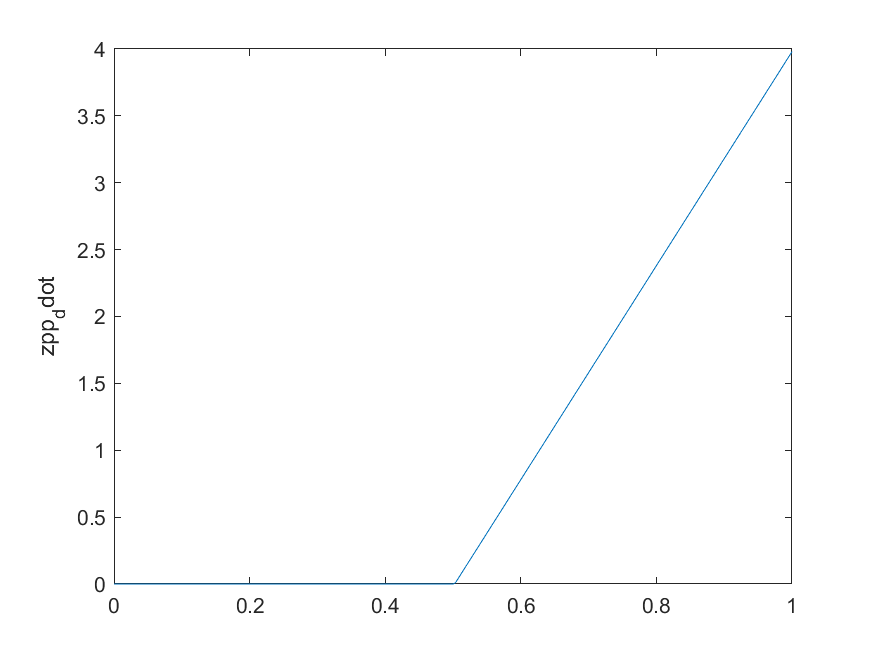
Raw data names are assumed to start with zpr\_

# 2. Fake data:

Fake data set is used to create a data set where damage is a simple function of strain (second order polynomial):



Corresponding to linear damage rate (dDot)



Note that for bulk damage models (sigma = (1 – D) sigma\_elastic, where sigma\_elastic = C epsilon)

Please see the file

zpp\_FakeDamageSet

It creates a raw data set:

rawData = gen\_textIndexedDatasets;

where as followed next, strain, stress, and optional damage, power terms are added to it.

# Config file

This is for how to process raw input data (zpr s) to get to processed data (critical points, homogenized damage model, …).

allRuns\_zppConfig = zpp\_DamageHomogenization1RConfig;

fidc = fopen('zpp\_config.txt', 'r');

Please open the file above. Some things specified in the files are:

1. Name of strain and stress fields in raw data (zpr …)
2. Threshold of homogenized damage, corresponding to initiation stage from the damage model.
3. The direction and sign of loading (e.g. tensile, compressive in xx are direction 1 with signs 1 and -1, shear is direction 3, …)
4. What are damage indicator names in raw data (if any). So, again raw data can also specify it’s own internal failure / degradation parameters which their nonzero level provides additional initiation points.
5. What are power (or integrand terms) in raw data that we want to integrate over time (or time steps). This field can be empty.

# Run map (if needed)

For a particular run, we can provide additional parameters that can affect going from raw strain / stress data to homogenized damage and critical points. These additional parameters are stored in a map that is in the form of a string key to a string value (string value is turned to a number is needed). An example is ( serialMax 399 ) that the user manuallry specifies to ignore whatever data is for this data set after runMap = gen\_map;

fidr = fopen('runMap.txt', 'r');

…

# Specifying stiffness matrix:

There are a couple of ways to do so. If the bulk is homogeneous, isotropic from elasticity perspective, we only need to specify E, nu, and plane mode (plane stress vs. plane strain). This is typically the case with aSDG runs where anisotropy / inhomogeneity is only in fracture strength (at least for now). For OSU data, the effect of particles in the matrix is having a potentially anisotropic material for each single sampled SVE. So, we somehow need to come up with the full C matrix for individual SVEs either from their linear analysis dataset OR by using their nonlinear analysis result say from the first few time steps.

# Normalization scales

Say raw data is specified in GPa but we want to present stresses in MPa, or other scenarios like this. In this case, we want to specify what is the scale of raw data quantities (relative to what we want them to be). The order of scales is:

U, V, stress, epsilon

For example for this fake data set I have used these scales

normalizationsUVSEpsIn = [0.0001 0.0862 0.1 0.1];

For OSU we probably can start with [1 1 1 1] for now.

# Raw data to processed data (zpr -> zpp)

This is done through these lines

zpp = zpp\_DamageHomogenization1R;

zpp = zpp.Initialize4DGRuns(allRuns\_zppConfig, runMap, E, nu, planeMode, normalizationsUVSEpsIn, t0, C);

zpp = zpp.Process(rawData);

# zpp\_DamageHomogenization1R class

This class is a bit more complex, and I leave it to you to go inside and see how it works.

But in short,

These are the names of the processed fields (serial number, time or serial value, scalar strain, scalar stress, strain tensor, stress tensor, Y (a parameter in damage homogenization), psi\_t, psi\_r, psi\_d, same 3 values normalized by total energy dissipation, homogenized damage parameter, and damage rate)

sn\_name = 'zpp\_sn';

time\_orsv\_name = 'zpp\_sv';

eps\_scalar\_name = 'zpp\_eps\_scalar';

sig\_scalar\_name = 'zpp\_sig\_scalar';

veps\_name = 'zpp\_eps\_voigt';

vsig\_name = 'zpp\_sig\_voigt';

Y\_name = 'zpp\_Y';

psi\_t\_name = 'zpp\_psi\_t';

psi\_r\_name = 'zpp\_psi\_r';

psi\_d\_name = 'zpp\_psi\_d';

psi\_t\_npsif\_name = 'zpp\_psi\_t\_npsif';

psi\_r\_npsif\_name = 'zpp\_psi\_r\_npsif';

psi\_d\_npsif\_name = 'zpp\_psi\_d\_npsif';

D\_name = 'zpp\_D';

Ddot\_name = 'zpp\_Ddot';

are the fields obtained from the analysis.

The corresponding values are put in this storage (named field storage):

pData = gen\_textIndexedDatasets;

It also stores critical points (cp\_), i.e. (order from last point on strain stress plot to first ones – initiation)

1. full damage (when strain is zero)
2. final point (the run may end before getting to stress = 0, in which case data for full damage is nan.
3. Max stress
4. Initiation based on homogenized damage >= provided tolerance (say 0.005)
5. Raw data provided failure parameters (e.g. crack length for aSDG runs and (damage and plasticity) for OSU results. There may be more than one raw damage indicator.

The names of these are:

cpt\_fullDamage\_AddedName = '\_cpt\_fulldamage';

cpt\_final\_AddedName = '\_cpt\_final';

cpt\_maxSts\_AddedName = '\_cpt\_maxsts';

cpt\_homogDini\_AddedName = '\_cpt\_homogdini';

cpt\_extDiniStart\_AddedName = '\_cpt\_damageext';

for example added raw data damage initiation number two is \_cpt\_damageext2

where are critical points stored? They are stored in:

1. cpt\_rawData = gen\_textIndexedDatasets; (a named map of critical points, but their raw data, zpr)
2. cpt\_pData = gen\_textIndexedDatasets; (a named map of critical points, but their processed data (e.g. damage, energies, etc.), zpp)
3. summaryCriticalVals\_rawData (summary of a)
4. summaryCriticalVals\_pData; (summary of b)

% summary of data is a matrix indexed as (criticalPt, fld) - these are in matrix format

# I/O operations for zpp class (zpp\_DamageHomogenization1R)

Once zpp data is created it can be stored and read back for future use, see:

nm = 'zzpProcessed';

fidzzc = fopen([nm, '.orig'], 'w');

zpp.toFile(fidzzc);

fclose(fidzzc);

zpp = cell(0);

zpp = zpp\_DamageHomogenization1R;

fidzzc = fopen([nm, '.orig'], 'r');

zpp = zpp.fromFile(fidzzc);

fclose(fidzzc);

fidzzc = fopen([nm, '2.orig'], 'w');

zpp.toFile(fidzzc);

fclose(fidzzc);

# Access of individual things in the processed data (zpp)

Consider the following scenarios for critical points:

1. Maximum stress [scalar]
2. All data corresponding to maximum stress stage (strain, damage, etc. at that stage) [vector]
3. All strains of different stages (eps\_failure, eps\_final, eps\_m, eps\_i, eps\_zpp damage indicators) [vector].
4. All critical points [matrix]

The set above, most likely is used for comparison between different runs. For example, how is the effect of volume ratio of inclusions on dissipated energy. For different runs, we go to their zpp and extract dissipated energy there, and plot all runs volume ratio of inclusion versus their corresponding dissipated energy (or say max damage).

OR histories (all time steps as opposed to only critical points) of zpp. Examples are:

1. Homogenized damage value (for all time steps)
2. Scalar strain
3. Dissipated energy
4. …

This time of data is most likely the time of plots we had in our Marble … paper. Different runs result in different cleaned-up strain stress responses (points beyond stress crossing zero are not shown). For example, refer to eps-sig plot in section 4 of that paper.

How is it done?

The simple call

data = zpp.getDataVectorByDataName(dataName);

returns a tensor (scalar, vector, matrix), where dataName is a specific name for any of the data types discussed above.

This block of the script file:

fidfld = fopen('\_testZPP.txt', 'r');

fidfldo = fopen('\_testZPP.out', 'w');

buf = READ(fidfld,'s');

buf = READ(fidfld,'s');

cntr = 0;

while (strcmp(buf, '}') == 0)

cntr = cntr + 1;

flds{cntr} = buf;

buf = READ(fidfld,'s');

end

fclose(fidfld);

for i = 1:length(flds)

dataName = flds{i};

data = zpp.getDataVectorByDataName(dataName);

fprintf(fidfldo, '\n\n%s\n', dataName);

gen\_toFile\_matrix(fidfldo, data);

end

fclose(fidfldo);

Opens a file with a bunch of names we want to check and prints them to \_testZPP.out.

So, I suggest that you open:

1. \_testZPP.txt (name of fields wanted to be viewed for this test case)
2. \_testZPP.out (the corresponding output tensor)

\_testZPP.txt has some comments in it, about how the names of all different tensors is constructed.