Data-Driven numerical site response

by J. Garcia-Suarez, A. Cornet, S. Wattel and J.-F. Molinari

This notebook contains computations leading to the results presented in the article “*data-driven numerical site response* ”.
Scripts to generate the original data (and the data itself) can be downloaded from *c4science.ch/-source/DD_1D-SRA*

Comparison to FEA

This portion implements the computations leading to Figure 6 in the text.

Start by setting the working directory (wherever the files, output of DEM and DDCM computations, “dataset.csv”, “displacements. csv” and “Amplification_Period. csv” are)

```
SetDirectory["..."];
```

Load data

```
In[2]:= data =
  Import["/home/joaquin/Desktop/Work/Student projects/Arthur Cornet/data/dataset.csv"];
```

Prepare data

We do some pre-processing to have the data in the right format.

```
In[3]:= strainList = {};
stressList = {};
Nelems = 20; (*because we discretize the column using 20 elements*)
Do[
 strain = Select[data[[All, 2 ii]], Developer`RealQ ];
 stress = Select[data[[All, 1 + 2 ii]], Developer`RealQ ];
 AppendTo[stressList , stress];
 AppendTo[strainList , strain];
 , {ii, 1, Nelems}]
```

Re-order for the colors

In[6]:=
```
order = Ordering[Max[#] & /@ stressList];
strainList = strainList[[order]];
stressList = stressList[[order]];
```

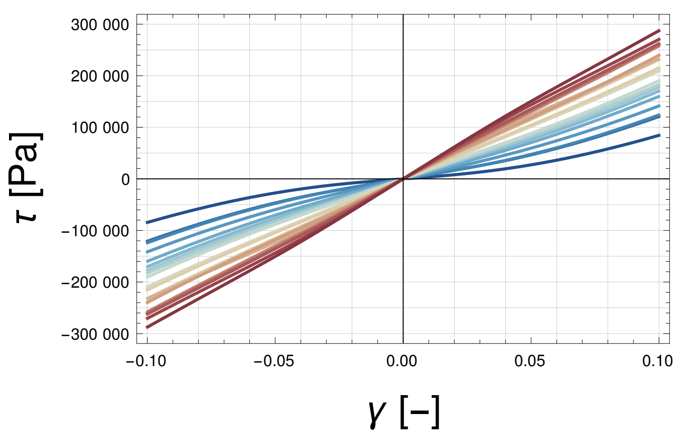
Prepare XY data

In[9]:=
```
auxData = Transpose @{strainList[[#]], stressList[[#]]} & /@ Range[20];
plotData = RandomChoice[#, 100] & /@ auxData;
```

Visualize the dataset (figure 5)

In[11]:=
```
fullColorList = ColorData["RedBlueTones"] /@ Subdivide[20];
ListPlot[
 auxData,
 Background → White,
 PlotStyle → Reverse[fullColorList],
 GridLines → All,
 Joined → True,
 Axes → Off,
 Frame → True,
 FrameStyle → Directive[Black],
 FrameLabel → {{Style["τ [Pa]", 20], None}, {Style["γ [–]", 20], None}}
]
```

Out[12]=



Sample to compute shear modulus

Pick the stress-strain data at a given level of strain (γ=0.05) to infer the shear stiffness:

In[13]:=
```
mus = #[[1500]][[2]] / #[[1500]][[1]] & /@ auxData;
```

In[14]:=
```
δl = 1 / 20.;
zs = Table[δl * (ii + 1/2), {ii, 0, -1 + Length @ mus}];
AppendTo[zs, 1.];
PrependTo[zs, 0.];
```

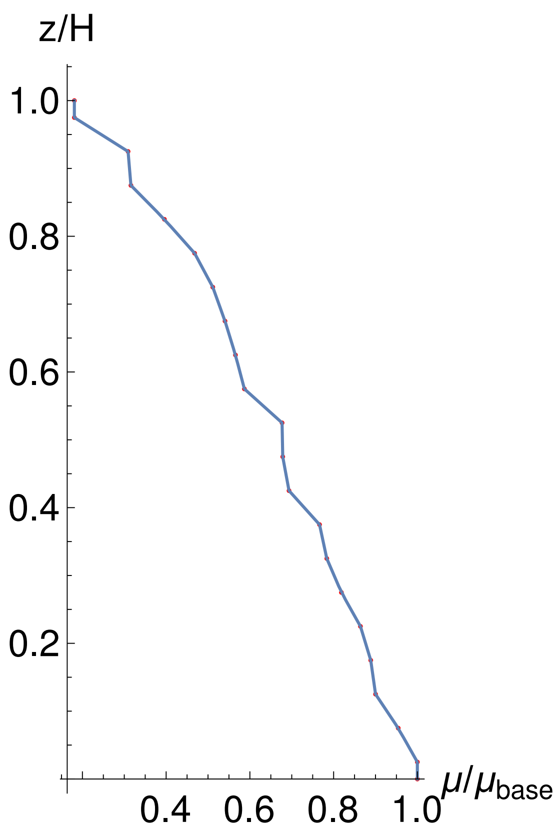
We need to provide the stiffness values over the whole domain, so we assign the first value in the list to the top surface and the last one to the bottom:

In[18]:=
```
extendedMus = Flatten @ Join[{mus[[1]], Sort[mus], mus[[-1]]}];
```

Next, we interpolate a function to pass to the solver (Figure 6a):

In[19]:=
```
muFunc = Interpolation [Transpose @{Reverse[zs], extendedMus }, InterpolationOrder  → 1];
Show[
 ListPlot[Transpose @{extendedMus / extendedMus [[-1]], Reverse @zs},
  Background → White,
  AspectRatio → 2,
  PlotStyle → Red,
  AxesStyle → Directive[Black, 20],
  AxesLabel → {"μ/μbase", "z/H"}],
 ParametricPlot [{muFunc[z] / muFunc[0], z}, {z, 0, 1}]
]
```

Out[20]=



Setting the FE analysis

In[21]:=
```
period = 0.1;(*base shake period [s]*)
timeStep = 0.001;(*time discretization [s]*)
T = period;
len = 1;(*domain size*)
(*PDE that we are to solve, equation(7) in the text,
the NeumannValue implements the stress-
 freee boundary condition at the top (see documentation below)*)
pde = 2500. * D[u[x, t], t, t] == D[muFunc[x] * D[u[x, t], x], x] + NeumannValue[0, x == 1];
(*initial condtions*)
ic = {u[x, 0] == 0, Derivative[0, 1][u][x, 0] == 0};
(*the other BC: forced displacement at the base*)
bc = u[0, t] == If[t < timeStep, 0, 1. * Sin[2 π * (t - timeStep)/T]];
```

In[28]:=
```
? NeumannValue
```

Out[28]=

Symbol

NeumannValue [*val*, *pred*] represents a Neumann boundary value *val*, specified on the part of the boundary of the region given to NDSolve and related functions where *pred* is True.

Solve equation (7) with appropriate initial and boundary conditions

In[29]:=
```
sol = NDSolveValue[{pde, ic, bc}, u, {t, 0, 3 * T}, {x, 0, 1},
   Method → {"TimeIntegration" → {"IDA", "MaxDifferenceOrder" → 2},
     "PDEDiscretization" →
      {"MethodOfLines", "DifferentiateBoundaryConditions" → True,
       "SpatialDiscretization" → {"FiniteElement", "InterpolationOrder" → {u → 2}}}
    }];
```

Compare to DD solution

Load the results of the DD analysis into the notebook:

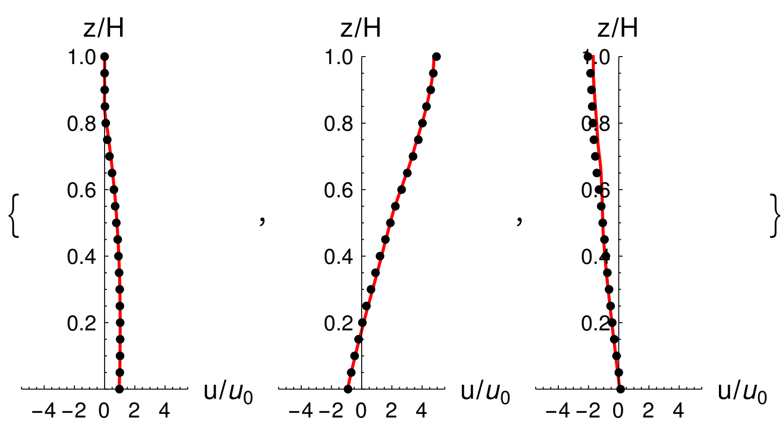
In[30]:=
```
data = Import["displacement.csv"];
dispTime = Table[Transpose@{data[[ii, ;;]]/0.01, 1/20 Range[0, 20, 1]}, {ii, 1, 301}];
```

Evaluate at the chosen timesteps and compare (Figure 6b):

In[32]:=
```
snaps = Table[
    ParametricPlot [{sol[x, ii * 10 timeStep], x}, {x, 0, 1},
     PlotRange → {{-5.5, 5.5}, {0, 1}},
     Background → White,
     Axes → True,
     AxesStyle → Directive[Black,  12],
     AxesLabel → {Style["u/u0", 15], Style["z/H", 15]},
     PlotStyle → Red,
     Epilog → {Black, PointSize[Medium], Point[{#[[1]], #[[2]]} & /@ dispTime[[10 * ii]]]},
     AspectRatio → 2,
     PlotRangeClipping → False,
     ImageSize → Small
    ]
   , {ii, {3, 17, 25}}];
```

In[33]:=
```
snaps
```

Out[33]=



Comparison to analytical

This second part implements the analytical transfer function assuming a linear evolution of stiffness, chosen to be represented by the data at γ=0.01, leading to Figure 7

Analytical solution

Let us verify equation(10) in the text. To do so, solve the dimensionless frequency-domain version of equation (7),
assuming $\mu(x) = \mu_{\text{base}}$(1-$\alpha$ x:)

In[34]:=
```
myODE = D[(1 - α * x) y '[x], x] + r^2 * y[x] == 0;(*ODE to solve*)
(*solve with stress-free BC at the top and fixed unitary amplitude at the base,
retrieve only the amplitude value at the top,
y[1], correspoding to the transfer function*)
sol = DSolveValue [{myODE , y[0] == 1, y '[1] == 0}, y[1], x];
```

In[36]:=
```
FullSimplify [sol](*simplify before displaying it*)
```

Out[36]=
$$\left(\sqrt{\frac{1}{1-\alpha}}\ \sqrt{1-\alpha}\right)\Big/\left(2\sqrt{\frac{r^2(-1+\alpha)}{\alpha^2}}\left(\text{BesselI}\left[1,\ 2\sqrt{\frac{r^2(-1+\alpha)}{\alpha^2}}\right]\text{BesselK}\left[0,\ 2\sqrt{-\frac{r^2}{\alpha^2}}\right]+\text{BesselJ}\left[0,\ \frac{2\,r}{\alpha}\right]\text{BesselK}\left[1,\ 2\sqrt{\frac{r^2(-1+\alpha)}{\alpha^2}}\right]\right)\right)$$

In[37]:=
```
Refine[%, 0 < α < 1](*further simplify it assuming α is positive and less than 1*)
```

Out[37]=
$$\alpha\Big/\left(2\sqrt{-r^2}\ \sqrt{1-\alpha}\left(\text{BesselI}\left[1,\ \frac{2\sqrt{-r^2}\ \sqrt{1-\alpha}}{\alpha}\right]\text{BesselK}\left[0,\ \frac{2\sqrt{-r^2}}{\alpha}\right]+\text{BesselJ}\left[0,\ \frac{2\,r}{\alpha}\right]\text{BesselK}\left[1,\ \frac{2\sqrt{-r^2}\ \sqrt{1-\alpha}}{\alpha}\right]\right)\right)$$

Assign this last value to a function **trFunc** to evaluate later:

In[38]:=
$$\text{trFunc}[\alpha\_,\ r\_] = \alpha\Big/\left(2\sqrt{-r^2}\ \sqrt{1-\alpha}\left(\text{BesselI}\left[1,\ \frac{2\sqrt{-r^2}\ \sqrt{1-\alpha}}{\alpha}\right]\text{BesselK}\left[0,\ \frac{2\sqrt{-r^2}}{\alpha}\right]+\text{BesselJ}\left[0,\ \frac{2\,r}{\alpha}\right]\text{BesselK}\left[1,\ \frac{2\sqrt{-r^2}\ \sqrt{1-\alpha}}{\alpha}\right]\right)\right);$$

Sample to compute shear modulus

Create a linear interpolation of the shear modulus data (at γ=0.01)

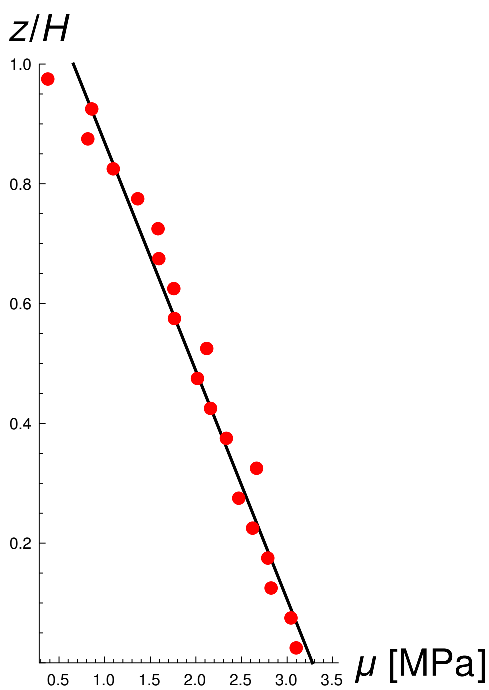
In[39]:=
$$\text{mus} = \frac{\#[[1100]][[2]]}{\#[[1100]][[1]]}\ \&\ /@\ \text{auxData};$$

In[40]:=
```
model = LinearModelFit [Transpose @{Reverse @ zs[[2 ;; -2]], mus}, x, x];
```

Visualize fit, Figure 7(b)

In[41]:=
```
ParametricPlot[{model["BestFit"] / 10^6, x}, {x, 0, 1},
 Background → White,
 AspectRatio → 2,
 Background → White,
 Axes → True,
 AxesStyle → Directive[Black],
 PlotRange → {{0.75 Min@mus / 10^6, 1.15 Max@mus / 10^6}, {0, 1}},
 PlotStyle → Black,
 AxesLabel → {Style["μ [MPa]", 20], Style["z/H", 20]},
 AspectRatio → 2,
 PlotRangeClipping → False,
 Epilog → {
   {Red, PointSize[Large], Point[Transpose@{mus / 10^6, Reverse@zs[[2 ;; -2]]}]}}
]
```

Out[41]=



Get the data from DDCM simulations

Load:

In[42]:=
```
data = Import["Amplification_Period .csv", "Table"];
```

Extract amplitudes and periods:

In[43]:=
```
amps = Flatten[ToExpression /@ StringSplit[data[[1]], ","]];
periods = Flatten[ToExpression /@ StringSplit[data[[3]], ","]];
```

Compute frequencies from periods:

In[45]:=
```
frequs = 1/# & /@ periods;
```

We also need the shear-wave velocity at the base as this parameter is used to write the aforementioned equation in dimensionless form

In[46]:=
```
Vbase = Sqrt[mus[[-1]] / 2500.];
```

Visualize comparison, Figure 7(b)

In[47]:=
```
DE = 0.07;
(*pick the coefficient of hysteretic
 damping to match the fundamental peak amplitude*)
LogLogPlot[{
  Abs[trFunc[1 - mus[[1]]/mus[[-1]], (2 π * f / Vbase)/Sqrt[1 + ⅈ * DE]]]
 }, {f, 1.1, 99.},
 Background → White,
 PlotRange → {{1, 100}, {0, 25}},
 GridLines → All,
 PlotStyle → Directive[Blue, Thickness[0.007]],
 PlotRange → All,
 Axes → False,
 Frame → True,
 FrameStyle → {{Thick, Black}, {Thick, Black}, {Thick, Black}, {Thick, Black}},
 FrameTicksStyle → 20,
 FrameLabel → {{Rotate[Style["û_top/û_base", 25], -π/2], None}, {Style["f[Hz]", 25], None}},
 GridLines → {All, None},
 PlotRangeClipping → False,
 Epilog → {{Black, PointSize[Medium], Point[Transpose @{Log[frequs], Log[amps]}]}}
]
```

Out[47]=