POLITECNICO
MILANO 1863

MSc. Music and Acoustic Engineering

Musical Acoustics - A.Y. 2020/2021

HL3 – Modeling Techniques

Authors' IDs:
10743504, 10751919,

January 16, 2021

1 Comments on the FD implementation

The MATLAB code `homework_piano.m` is able to perform a backward finite difference (FD) scheme which is used to simulate a piano string struck with a felt hammer. In the first part of the following report the numerical values chosen are illustrated while the second part describes briefly the actual FD algorithm. Finally the last section shows some results of such implementation.

1.1 Simulation data

The string under consideration is tuned to C2 ($f_0 = 52.8221\,\text{Hz}$). We chose a sampling frequency of $f_s = 176.4\,\text{kHz}$ as suggested in [1] and it's well above the Nyquist frequency so we don't expect aliasing in the time domain. Other dimensions that are not suggested in the assignment, such as string mass, hammer mass, string length etc. have been taken from [1] and can be found listed in Tab. 1 . The string has a linear mass $\mu = M_s/L$ and the tension applied on it at rest has been computed as $T_0 = 4L^2 f_0^2 \mu$ which determines that the speed of sound along the string is $c = \sqrt{T_0/\mu}$. The spatial sampling has been chosen under the limit imposed by the stability condition [2]

$$N_{max} = \sqrt{\frac{-1 + \sqrt{1 + 16\epsilon\gamma^2}}{8\epsilon}}$$

where $\gamma = f_s/2f_0$. The number of spatial steps has been set as $N = N_{max} - 1 = 537$ and the relative spatial resolution is $X = 0.0036\,\text{m}$.

string length	L	$1.92\,\text{m}$
string mass	M_s	$35 \times 10^{-3}\,\text{kg}$
string stiffness parameter	ϵ	7.5×10^{-6}
string stiffness coefficient	κ	7.5×10^{-6}
relative striking position	a	0.12
viscous damping coefficient	b_h	$1 \times 10^{-4}\,\text{s}^{-1}$
hammer mass	M_h	$4.9 \times 10^{-3}\,\text{kg}$
hammer stiffness	K	$4 \times 10^{8}\,\text{N}\,\text{m}^{-1}$
stiffness exponent	p	2.3
hinge normalized impedance	ζ_l	$10^{20}\,\text{s}^3\text{m}^2/\text{kg}^2$
bridge normalized impedance	ζ_b	$10^{3}\,\text{s}^3\text{m}^2/\text{kg}^2$

Table 1: Values considered in the simulation.

1.2 FD algorithm

The definition of temporal and spatial coordinate is the very first step of the FD implementation

```
t = linspace(0, duration, tSamples); %temporal vector
x = linspace( 1 , L , L/X  ); %spatial coordinate vector
```

followed by the initialization of the string displacement $y(n, m)$, the hammer displacement (with respect to the contact point) $\eta(n)$, the hammer force $F(n)$ and the Hanning window $g(m)$, where $n \in t$ and $m \in x$.

```
y = zeros(length(t), length(x)); %string displacement vector
eta = zeros(1 , length(t)); %hammer displacement vector
Fh = zeros(1 , length(t)); %hammer force vector
g = zeros(length(x), 1 ); %Hanning window vector
```

At this point the Hanning window must be defined centred in the striking point m_0 and with a width of $2w$

```
m0 = ceil(x0/X); % hammer striking coordinate
wh = 0.1; %[m] half width of the hammer
w = ceil(wh/X); %sample of the hammer's width
gloc = hann(2*w); %local hanning window
g(m0-w:m0-w+length(gloc)-1)=gloc;
```

and used to filter the force that is acting on the string $F(n, m) = F(n)g(m)$. At time $t = 0$ $(n = 1)$ the string and the hammer are supposed to be still $y(1, m) = \eta(1) = 0$ and therefore the force $F(1) = 0$, since it is defined by the non-linear relation $F(n) = K|\eta(n) - y(n, m0)|^p$. The hammer displacement and the force at time $t = dt$ $(n = 2)$ is given by the velocity at time $t = 0$, which is $Vh_0 = 2.5\,\mathrm{m\,s^{-1}}$

```
eta(2) = Vh0*ts;
Fh(2) = K*(abs(eta(2) - y(2, m0)))^p;
```

where $y(2, m_0)$ is assumed to be zero. After having defined the initial conditions, the backward FD loop can start by computing the displacement at time sample $n + 1$ by knowing the displacement at n and $n - 1$.

The scheme proposed below represents an implementation of the algorithm described in [1], where boundary conditions take into account the force transmission at the hinge and at the bridge, the displacement approximation at points just inside the domain ($x = dx$ and $x = L - dx$) and the hammer displacement at any time step. The FD formulas contain constant and convenient parameters which have not been mentioned before but can be found in [1].

```
for n = 2:length(t)-1

Fh(n) = K*(abs(eta(n) - y(n, m0)))^p; % stricking force

for m = 1:length(x) %along the string

%forcing term definition at any time step

%if the hammer is no more in contact with the string
if(eta(n) < y(n, m0))
Fh(n) = 0;

else
```

```
F(n,m) = Fh(n)'*g(m); %windowed with the hanning
end

%boundary conditions
if(m==1) %if x = 0
y(n+1, m) = bl1*y(n, m) + bl2*y(n, m+1) + bl3*y(n, m +2) + ...
bl4*y(n-1, m) + blf*F(n,m);

elseif (m== length(x)) % if x = L
y(n+1, m) = br1*y(n, m) + br2*y(n, m-1) + br3*y(n, m -2) + ...
br4*y(n-1, m) + brf*F(n,m);

elseif( m ==length(x)-1 ) % if x = L - dx
y(n+1,m) = a1*(2*y(n, m+1) - y(n,m) + y(n,m-2)) + a2*(y(n,m+1) + ...
y(n,m-1)) + a3*y(n,m) + a4*y(n-1,m) + a5*(y(n-1,m+1) + y(n-1,m-1)) + ...
af*F(n,m);

elseif( m == 2) %if x = dx
y(n+1,m) = a1*(y(n, m +2) - y(n,m) + 2*y(n,m-1)) + a2*(y(n, m+1) ...
+ y(n,m-1)) + a3*y(n,m) + a4*y(n-1,m) + a5*(y(n-1,m+1) + ...
 y(n-1,m-1)) + af*F(n,m);

else
%backward finite difference scheme "inside the domain"
y(n+1,m) = a1*(y(n, m+2) + y(n, m-2)) + a2*(y(n, m+1) +...
 y(n, m-1)) + ... a3*y(n,m) + a4*y(n-1,m) + ...
  a5*(y(n-1, m+1) + y(n-1,m-1)) + af*F(n,m);

end
end

%hammer displacement approximation
eta(n+1) = d1*eta(n) + d2*eta(n-1) + df*Fh(n);
end
```

It can be noticed that the force $F(n)$ becomes zero when the hammer is no longer in contact with the string, so when $\eta(n) < y(n, m_0)$.

1.3 Time history and audio file

After the vector $y(n, m)$ has been computed for every value of n and m with the previous scheme, a visual representation can be obtained by plotting the displacement along x for each time step. Actually, since in this way the animation would be too slow, it has been decided to plot $y(n, m)$ each 50 time steps. A snapshot is shown in Fig. 1 and an animation called `string_displacement.gif` can be found attached to this report.

Moreover, a mean value of the displacement around the point $r_0 = L - m0$ (see Fig.2 and Fig.3) has been computed by averaging 12 spatial samples: 6 on the left of r_0 and 6 on the right of r_0

```
r0 = ceil((L-x0)/X); %coordinate of the striking specular point
yspec = [y(:,r0+1)' ; y(:,r0+2)'; y(:,r0+3)'; y(:,r0+4)'; ...
 y(:,r0+5)'; y(:,r0+6)'; y(:,r0-1)'; y(:,r0-2)'; ...
 y(:,r0-3)'; y(:,r0-4)'; y(:,r0-5)'; y(:,r0-6)'];
yaudio = mean(yspec);
```

from which the vector `yaudio` has been amplified by a factor of $\times 100$ in order to make it audible, then converted into a sound and stored it.

```
sound(100*yaudio, fs); %Play the sound
 filename = '10751919_Lercari_10743504_Lampis_piano.wav';
audiowrite(filename, 100*yaudio,fs);
```

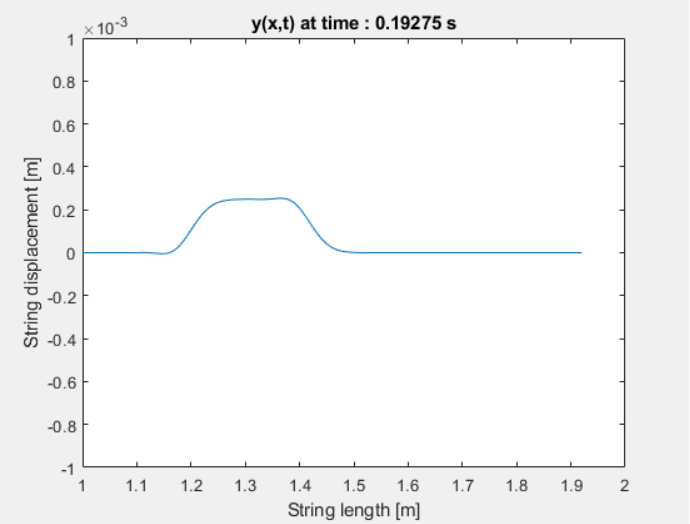

Figure 1: Snapshot from string displacement animation.

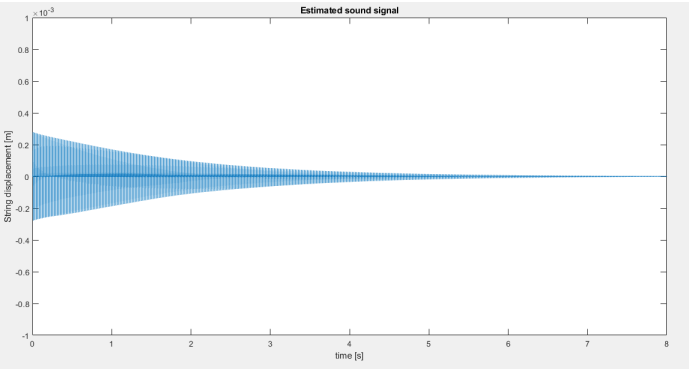

Figure 2: Time history of the mean of the string displacement.

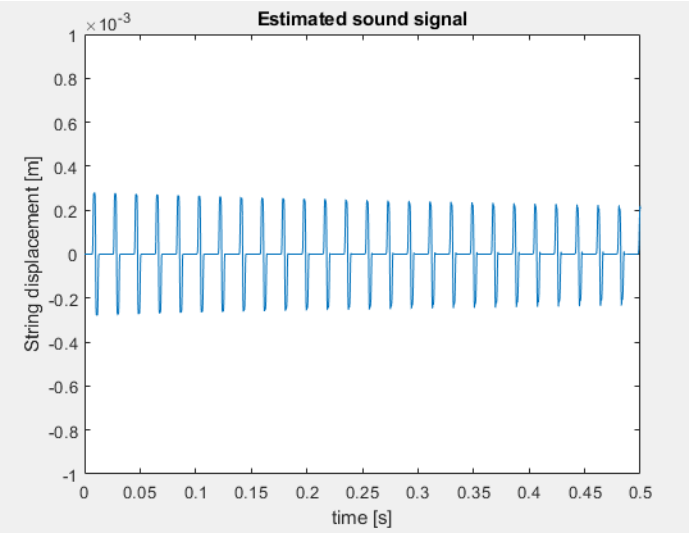

Figure 3: Time history of the mean of the string displacement during the first 500 ms.

2 Modeling a guitar with an electrical analog

2.1 Circuit for the body

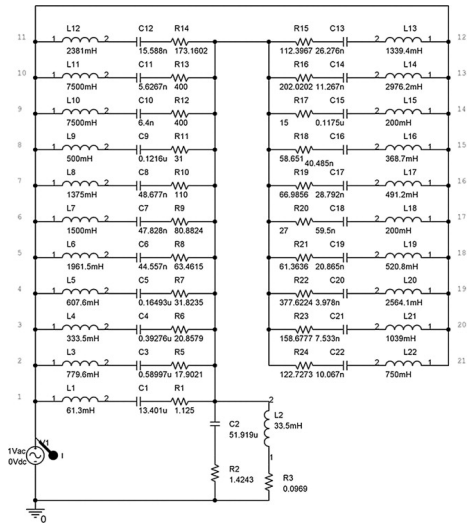

Figure 4: Network analog of the guitar body, constructed by expanding on the two-mass model.

The simplest model of the body of a guitar is the two-mass model, which corresponds to the circuit in Fig. 4 when only the first of the many RLC series is present. This is a lumped parameter model which accounts well for the lowest modes of the guitar body. If we want to include higher modes in the model using

theoretical arguments, we are going to need some kind of distibuted-parameter component, at least for the top plate. Alternatively, we can follow an empirical approach such as the one presented in [3]. Here, 20 additional resonances of the top plate are modelled as RLC series in parallel with the first one, expanding on the two-mass model. The components are then tuned to make the circuit response resemble that of an actual instrument (as measured by the authors). This approach allows us to use a representation based only on discrete, passive components, simplifying the implementation significantly and granting us the possibility of using one of the many available tools for the simulation of analog electrical networks. In particular, the implementation we present in this section has been done with Simscape.

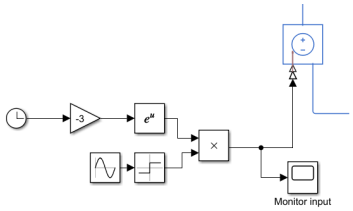

Figure 5: Simplified model for the input signal at the bridge.

In the first part of the implementation we are concerned with designing the body circuit. Here we run a simulation ignoring, for the moment, the modeling of the string: to do so we use a rough approximation for the string signal, given by a damped square wave (Fig. 5). The body network was constructed using the component values shown in Fig. 4 and it can be found in `exercise_6a.slx`. The resulting signal can be seen in Fig. 6, and it has been saved as an audio file in `simplifiedInput.wav`.

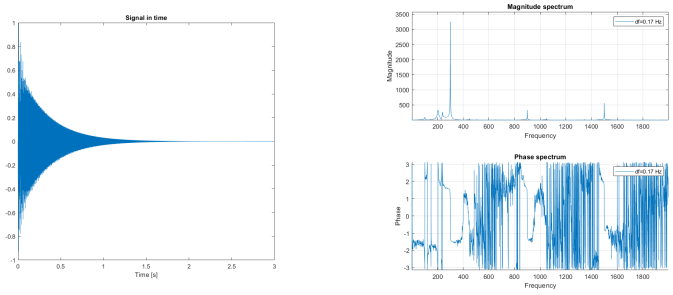

Figure 6: Time history and FFT of the (normalized) sound with the simplified input.

2.2 Circuit for the string (and the whole guitar)

The second part of the implementation involves instead the modeling of the string. The string itself can be represented by a transmission line, with the ideal reflections at the ends corresponding to short circuits. In parallel with the short circuits, we can see at both ends a current generator with a load impedance, used to model the plucking of the string. The generators provide the initial conditions by feeding triangular inputs to the transmission line. After half a period of the fundamental, the generator switches open and the short-circuit switches close, leaving the signal to reflect back and forth on the transmission line.

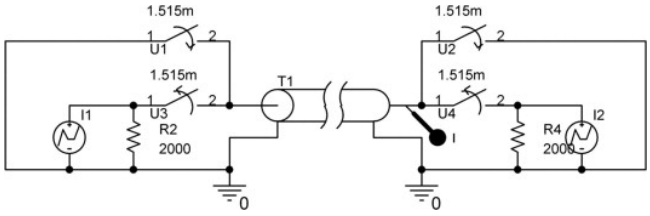

Figure 7: Electrical analog of a guitar string.

The string circuit is connected to the body circuit at the U2 switch (Fig. 7. The sound is recovered by taking the time derivative (approximated with finite differences) of the current at this point, which can be shown to be proportional to the vertical force at the bridge. The resulting signal can be seen in Fig. 8 and it has been saved as an audio file in `completeGuitar.wav`. The MATLAB script that executes the simulation is `exercise6_script.m`, while the complete circuit model can be found in `exercise_6.slx`. The result is a much better approximation of the sound of a guitar than the two-mass system, and this is especially apparent in how the fast plucking transient sounds, as a consequence of the more realistic response in the higher frequency range.

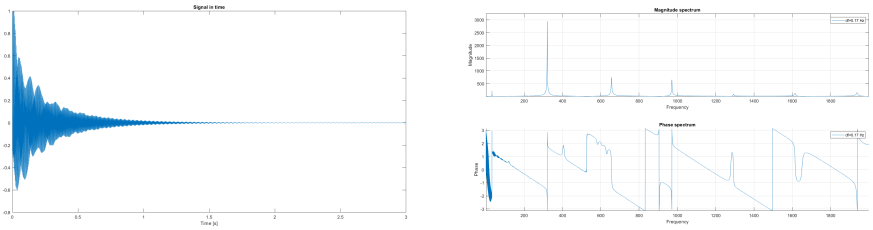

Figure 8: Time history and FFT of the (normalized) sound produced by the complete guitar simulation.

References

[1] Charalampos Saitis, Sarah Orr, and Maarten Van Walstijn. “Physical modeling of the piano: An investigation into the effect of string stiffness on the hammer string interaction.” In: *The Journal of the Acoustical Society of America* 125 (May 2009), p. 2684. DOI: 10.1121/1.4784255.

[2] Antoine Chaigne and Anders Askenfelt. “Numerical simulations of piano strings. I. A physical model for a struck string using finite difference methods”. In: *Journal of The Acoustical Society of America - J ACOUST SOC AMER* 95 (Feb. 1994), pp. 1112–1118. DOI: 10.1121/1.408459.

[3] Jingeol Lee and Mark French. “Circuit based classical guitar model”. In: *Applied Acoustics* 97 (2015), pp. 96–103. ISSN: 0003-682X. DOI: https://doi.org/10.1016/j.apacoust.2015.04.006. URL: http://www.sciencedirect.com/science/article/pii/S0003682X15001152.