Analysis of a vibrating string under tension with XC

Figure 1: Waves on a harp *(retrieved from www.dslrvideocollege.com)*

Ana Ortega, Luis C. Pérez Tato

Introduction

XC is a finite element program oriented to civil engineering. It is conceived as Open Source Software since we are developing it on the strong foundations of *OpenSees* and making heavy use of other OSS like Python, VTK and CGAL.
This case study deals with a flexible string fixed at the ends and stretched to an initial strain. The aim is to determine the first three natural frequencies of lateral vibration of the stretched string.

Theoretical introduction

Any motion of a linear system can be resolved into a superposition of modes, combined with appropriate amplitudes and phases [2]. Modes are special pattern of motions that have the property that at any point the object moves perfectly sinusoidally and that all points move at the same frequency (if conditions, e.g. stiffness, boundaries, ..., do not change in time).

Our example deals with a one-dimensional wave confined in the two ends of a string. The general solution for the motion is the sum of two functions, $F(x-ct)$ and $G(x+ct)$, the first representing a wave travelling one way in the string, and the second a wave travelling the other way in the string:

$$y = F(x-ct) + G(x+ct) \tag{1}$$

We have to satisfy the conditions that the string does not move at both ends. If we firstly put $x=0$ in Eq. 1, we get $y = F(-ct) + G(ct)$ and, if this is to be zero for all times, it means that $G(ct)$ must be $-F(-ct)$. Putting back this result into Eq. 1, we find

$$y = F(x-ct) - F(-x-ct) \tag{2}$$

The Eq. 2 found for the total motion of the string can be regarded as the sum of two waves in the region of positive x, as shown in figure 2. The first wave travels in the negative x-direction, and the second one (hypothetical) travels in the other direction, reversed in sign and on the other side of the origin. As they reach the origin, they always cancel at x=0, and finally the second (reflected) wave will be the only one to exist for positive x and it will, be travelling in the opposite direction.

If the wave is periodic, Eq. 2 can be expressed as:

$$\begin{aligned} y = F(x-ct) - F(-x-ct) = \\ e^{i\omega(t-x/c)} - e^{i\omega(t+x/c)} = \\ -2ie^{i\omega t}\sin(\omega x/c) \end{aligned} \tag{3}$$

In this solution, for any point x, the string oscillates at the same frequency ω. Points with no motion, called

nodes, must satisfy the condition $\sin(\frac{\omega x}{c}) = 0$, which means that $\frac{\omega x}{c} = 0, \pi, 2\pi, \cdots, n\pi, \cdots$.

In our case, the string is held at both ends, say at x=0 and x=L. For having a periodic sinusoidal motion, the only possibility is that the sine wave must just fit into the string length; in that case, it will continue to keep that perfect shape of a sine wave and will oscillate harmonically at some frequency.

We can write $y = \sin\left(\frac{\omega x}{c}\right)$ for the shape. To keep both ends fixed, it must be $\sin\left(\frac{\omega L}{c}\right) = 0$ and therefore $\frac{\omega L}{c} = n\pi$.

So, the string can have sinusoidal motions, but only at certain frequencies,

$$\omega = \frac{n\pi c}{L}$$

that are a property of the particular system and the nature of its boundaries

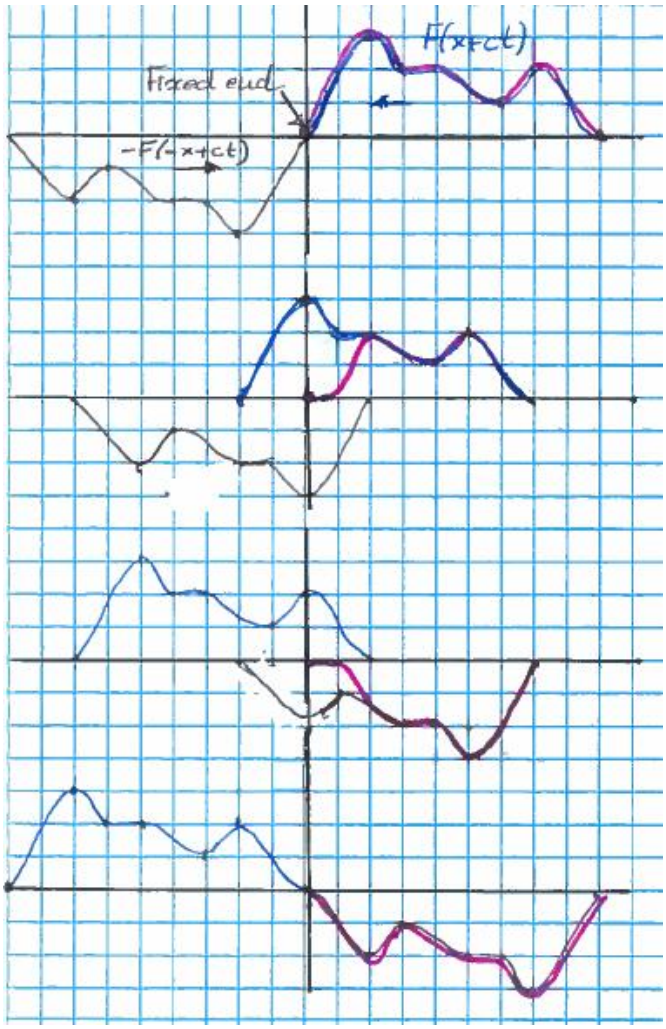

Figure 2: Reflection of a wave

Modes of vibration of a string under tension

Let's say a flexible string of mass ρ per unit length is stretched under tension T. By assuming the lateral deflection y of the string to be small, the change in tension with deflection is negligible and can be ignored.

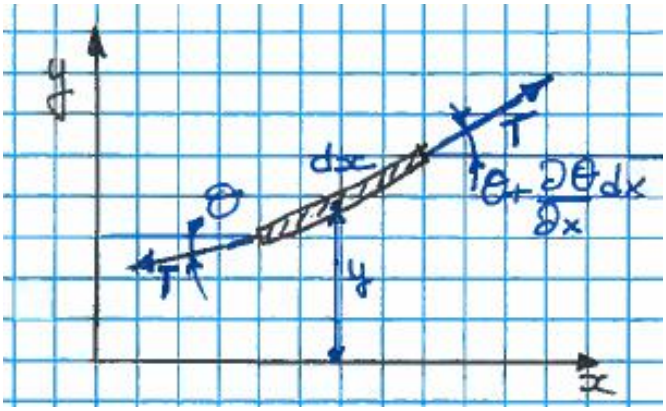

Figure 3: String element under tension

In Fig. 3 a free-body diagram of an elementary length dx of the string is shown. Assuming small deflections and slopes, the equation of motion in y direction is

$$\begin{aligned} F_y &= m \cdot a_y \\ T(\theta + \frac{\partial \theta}{\partial x}dx) - T\theta &= \rho dx \cdot \frac{\partial^2 y}{\partial t^2} \\ T \cdot \frac{\partial \theta}{\partial x}dx &= \rho dx \cdot \frac{\partial^2 y}{\partial t^2} \\ \frac{\partial \theta}{\partial x} &= \frac{\rho}{T} \cdot \frac{\partial^2 y}{\partial t^2} \end{aligned} \tag{4}$$

Since the slope of the string is $\theta = \frac{\partial y}{\partial x}$, the equation 4 reduces to

$$\frac{\partial^2 y}{\partial x^2} = \frac{1}{c^2} \cdot \frac{\partial^2 y}{\partial t^2} \tag{5}$$

where $c = \sqrt{T/\rho}$ can be shown to be the velocity of wave propagation along the string.

The equation 5 can be solved by separation of variables, assuming the solution in the form:

$$y(x,t) = Y(x) \cdot G(t)$$

By substitution into Eq.5, we obtain:

$$\begin{aligned} G \cdot \frac{d^2 Y}{dx^2} &= \frac{1}{c^2} \cdot Y \cdot \frac{d^2 G}{dt^2} \\ \frac{1}{Y} \cdot \frac{d^2 Y}{dx^2} &= \frac{1}{c^2} \cdot \frac{1}{G} \cdot \frac{d^2 G}{dt^2} \end{aligned} \tag{6}$$

Since the left side of this equation is independent of t, whereas the right side is independent of x, each side must be a constant. Letting this constant be equal to $-\left(\frac{\omega}{c}\right)^2$,

we obtain two ordinary differential equations

$$\begin{aligned} \frac{d^2Y}{dx^2} + \left(\frac{\omega}{c}\right)^2 \cdot Y &= 0 \\ \frac{d^2G}{dt^2} + \omega^2 \cdot G &= 0 \end{aligned} \tag{7}$$

with the general solutions

$$\begin{aligned} Y &= A \cdot \sin(\frac{\omega}{c}x) + B \cdot \cos(\frac{\omega}{c}x) \\ G &= C \cdot \sin(\omega t) + D \cdot \cos(\omega t) \end{aligned} \tag{8}$$

The arbitrary constants A, B, C, D depend on the boundary conditions and the initial conditions. In the case of a string stretched between two fixed points separated a distance L, the boundary conditions are $y(0,t) = y(L,t) = 0$. The condition that $y(0,t) = 0$ requires that $B = 0$, and

$$y = (C \cdot \sin(\omega t) + D \cdot \cos(\omega t)) \cdot \sin(\frac{\omega}{c}x) \tag{9}$$

The condition $y(L,t) = 0$ requires that $\sin(\frac{\omega L}{c}) = 0$, or

$$\frac{\omega_n L}{c} = \frac{2\pi L}{\lambda} = n\pi, \quad n = 1, 2, 3, \cdots \tag{10}$$

where $\lambda = \frac{c}{f}$ is the wavelength and f is the frequency of oscillation. Each n represents a normal mode vibration with natural frequency determined from the equation

$$f_n = \frac{n}{2L}c = \frac{n}{2L}\sqrt{\frac{T}{\rho}}, \quad n = 1, 2, 3, \cdots \tag{11}$$

The mode shape is sinusoidal with the distribution

$$Y = \sin\left(n\pi\frac{x}{L}\right) \tag{12}$$

Test case solved with XC

A uniform steel string of length $L = 2m$ is fixed at the ends and stretched to an initial strain $\epsilon_0 = 0.005$. Its geometric and material properties are represented in figure 4. The goal is to determine its first three natural frequencies of lateral vibration.

$E_{steel} =$	2.1e11	Pa
$\rho_{steel} =$	7850	kg/m^3
$L_{string} =$	2.0	m
$A_{cross\ section} =$	2.0e-6	m^2
$\epsilon_0 = 5‰$		

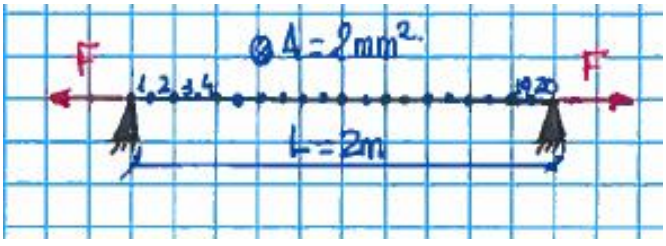

Figure 4: String, geometric and material properties

Under these conditions, the stress and force in the string will be:

$$\sigma = E_{steel} \times \epsilon_0 = 1050e6Pa$$

$$F = \sigma \times A_{cross\ section} = 2100.0N$$

and the mass per unit length:

$$\rho_{string} = \rho_{steel} \times A_{cross\ section} = 0.0157kg/m$$

By applying Eq. 11 for n= 1,2,3 with $c = \sqrt{\frac{T}{\rho_{string}}} = 365.7\ m/s$, we obtain the exact value of the first three frequencies:

$$\begin{aligned} f_1 &= \frac{1}{2L}c = 91.432Hz \\ f_2 &= \frac{2}{2L}c = 182.865Hz \\ f_3 &= \frac{3}{2L}c = 274.297Hz \end{aligned}$$

XC model

The input python script that solves the present case in XC can be found at the end of this document.

To build the FE model, two points at the ends of the string are created and linked by a line. A material of type *cable* is defined, giving as input the elastic modulus, prestress and mass per unit length. The element used to mesh the line is of type corotational truss. Nodes in the ends of the string are constrained in all of their degrees of freedom.

The following step is to define the analysis type and its options.

XC inherit from *OpenSees* the objects responsible for performing the analysis. For a general transient dynamic analysis, the basic equation of motion to be solved is

$$[M]\{\ddot{u}\} + [C]\{\dot{u}\} + [K]\{u\} = \{F(t)\} \tag{13}$$

where,

[M] = mass matrix

[C] = damping matrix

[K] = stiffness matrix

$\{\ddot{u}\}$ = nodal acceleration vector

$\{\dot{u}\}$ = nodal velocity vector

$\{u\}$ = nodal displacement vector

$\{F(t)\}$ = load vector

The component classes involved in the analysis consist of the following:

ConstraintHandler determines how the constraint equations are enforced in the analysis, that is, how it handles the boundary conditions and imposed displacements.

DOF_Numberer determines the mapping between equation numbers and degrees of freedom, or how the DOF are numbered.

AnalysisModel defines the type of analysis to be performed.

Integrator determines the predictive step for time $t + dt$.

SolutionAlgorithm determines the sequence of steps taken to solve the non-linear equation at the current time step

SystemOfEqn/Solver within the solution algorithm, it specifies how to store and solve the sytem of equations in the analysis

Firstly, a static analysis is performed. The load is applied in 10 steps and the static equilibrium is obtained iteratively in multiple steps using the Newton-Raphson solution algorithm. The prestressing force is obtained for the elements.

Next, a modal analysis is performed to get the first eigenvalues.

In this problem, we have only homogeneous single-point boundary constraints, this is why the plain handler is used as ConstraintHandler. For other non-homogeneus constraints, like imposed motions, multi support excitation, etc, penalty, Lagrange multipliers and transformation methods are available.

For this small problem, a *plain* numberer is used to assign the DOF to the nodes; for larger models the algorithm *RCM (Reverse Cuthill-McKee)*, that optimizes node numbering and reduces bandwidth, could be a preferable option. A symmetric positive definite banded system of equations is factored and solved during the analysis.

By performing a frequency analysis using the eigen solver, the three natural frequencies were calculated as 91.338, 182.114 and 271.766 Hz respectively, which approximate the target values with an accuracy greater than 99%.

Quantity	**Target**	**XC result**	**Ratio**
σ	1050 MPa	1049.999 MPa	≈ 1.000
F	2100.0 N	2099.999 N	≈ 1.000
f_1	91.432 Hz	91.338 Hz	0.999
f_2	182.865 Hz	182.114 Hz	0.996
f_3	274.297 Hz	271.766 Hz	0.991

References

[1] Silvia Mazzoni Frank McKenna Michael H. Scott Gregory L. Fenves et al. Opensees command language manual. Technical report, Earthquake Engineering Research Center. College of Engineering. University of California, Berkeley, 2006.

[2] Sands Feynman, Leighton. *The Feynman Lectures on Physics, Volume I.* Caltech.

[3] William T. Thomson. *Theory of vibration with applications, second edition.* George Allen & Unwin.

```
# -*- coding: utf-8 -*-
''' Calculation of the first three natural frequencies of lateral vibration
    in a stretched string fixed at both ends.
    Reference: Theory of vibration with applications, by William T. Thomson.
               Second edition, app. 7.1
   '''
from __future__ import division
import xc_base
import geom
import xc

from model import fix_node_3dof
from model import fix_nodes_lines
from model import predefined_spaces
from materials import typical_materials
import math

NumDiv= 20              # number of elements
E= 2.1e11               # Young modulus [Pa]
l= 2.0                  # length of the vibrating string [m]
epsilon=0.005           # initial elongation
area= 2e-6              # cross-sectional area [m2]
mDens=7850              # mass density [kg/m3]
Mass= mDens*area        # mass per unit length [kg/m]
sigmaPret= E*epsilon    # prestressing stress [Pa]
fPret= sigmaPret*area   # prestressing force [N]


#      * Model *
FEcase= xc.ProblemaEF()
preprocessor=  FEcase.getPreprocessor
nodes= preprocessor.getNodeLoader

# Problem type
predefined_spaces.gdls_resist_materiales2D(nodes)
nodes.newSeedNode()

# Materials definition
typical_materials.defCableMaterial(preprocessor, "cable",E,sigmaPret,Mass)

# Seed element
seedElemLoader= preprocessor.getElementLoader.seedElemLoader
seedElemLoader.defaultMaterial= "cable"
seedElemLoader.dimElem= 2
seedElemLoader.defaultTag= 1 #Tag for the next element.
truss= seedElemLoader.newElement("corot_truss",xc.ID([0,0]))
truss.area= area

# Points and lines
points= preprocessor.getCad.getPoints
pt= points.newPntIDPos3d(1,geom.Pos3d(0.0,0.0,0.0))
pt= points.newPntIDPos3d(2,geom.Pos3d(1,0.0,0.0))
lines= preprocessor.getCad.getLines
lines.defaultTag= 1
l= lines.newLine(1,2)
l.nDiv= NumDiv

l1= preprocessor.getSets.getSet("l1")
l1.genMesh(xc.meshDir.I)

# Constraints
coacciones= preprocessor.getConstraintLoader
fix_nodes_lines.ConstraintsForLineExtremeNodes(l1,coacciones,fix_node_3dof.fixNode000)
fix_nodes_lines.ConstraintsForLineInteriorNodes(l1,coacciones,fix_node_3dof.fixNodeFF0)

#    *Static analysis*
Nstep= 10       #  apply load in 10 steps
DInc= 1./Nstep  #  first load increment
solu= FEcase.getSoluProc
solCtrl= solu.getSoluControl
solModels= solCtrl.getModelWrapperContainer
sm= solModels.newModelWrapper("sm")
```

```
cHandler= sm.newConstraintHandler("plain_handler")
numberer= sm.newNumberer("default_numberer")
numberer.useAlgorithm("simple")
solMethods= solCtrl.getSoluMethodContainer
smt= solMethods.newSoluMethod("smt","sm")
solAlgo= smt.newSolutionAlgorithm("newton_raphson_soln_algo")
ctest= smt.newConvergenceTest("norm_unbalance_conv_test")
ctest.tol= 1e-8
ctest.maxNumIter= 100
integ= smt.newIntegrator("load_control_integrator",xc.Vector([]))
integ.dLambda1= DInc
soe= smt.newSystemOfEqn("band_gen_lin_soe")
solver= soe.newSolver("band_gen_lin_lapack_solver")
analysis= solu.newAnalysis("static_analysis","smt","")
result= analysis.analyze(Nstep)

#Results
elementos= preprocessor.getElementLoader
ele1= elementos.getElement(1)
prestrF= ele1.getN()
sigma= ele1.getMaterial().getStress()


#       *Modal analysis*
solCtrl= solu.getSoluControl
solModels= solCtrl.getModelWrapperContainer
sm= solModels.newModelWrapper("sm")
cHandler= sm.newConstraintHandler("plain_handler")
numberer= sm.newNumberer("default_numberer")
numberer.useAlgorithm("simple")
solMethods= solCtrl.getSoluMethodContainer
smt= solMethods.newSoluMethod("smt","sm")
solAlgo= smt.newSolutionAlgorithm("frequency_soln_algo")
integ= smt.newIntegrator("eigen_integrator",xc.Vector([1.0,1,1.0,1.0]))
soe= smt.newSystemOfEqn("sym_band_eigen_soe")
solver= soe.newSolver("sym_band_eigen_solver")
analysis= solu.newAnalysis("eigen_analysis","smt","")

analOk= analysis.analyze(3)

#Results
eig1= analysis.getEigenvalue(1)
eig2= analysis.getEigenvalue(2)
eig3= analysis.getEigenvalue(3)
f1= math.sqrt(eig1)/(2*math.pi)
f2= math.sqrt(eig2)/(2*math.pi)
f3= math.sqrt(eig3)/(2*math.pi)



print "Stress= ",sigma
print "Prestressing force= ",prestrF

print "eig1= ",eig1
print "eig2= ",eig2
print "eig3= ",eig3
print "f1= ", f1
print "f2= ", f2
print "f3= ",f3
```