The Journal of Open Source Software

bridge-sim: A Python package for concrete slab bridge simulation

Jeremy Barisch-Rooney[1, 2]

1 TNO **2** University of Amsterdam

DOI: 10.21105/joss.0XXXX

Software

- Review
- Repository
- Archive

Editor: Editor Name

Submitted: 01 January 1900
Published: 01 January 3030

Summary

`bridge-sim` is an open source package for Python that provides a high-level API for building linear and non-linear 3D models of concrete slab bridges, running simulations of the generated models with OpenSees (McKenna, 2011), and generating time series and graphics of the responses e.g. Figure 1. `bridge-sim` has been written with extensibility in mind such that adding support for another finite element program other than OpenSees is possible.

`bridge-sim` makes the process of generating time series data via simulation of concrete slab bridges much faster and easier than with OpenSees – OpenSees has no knowledge of concepts such as briges, traffic or temperature. `bridge-sim` accomplishes this by providing a class-based API with classes such as `Bridge`, `Material` and `Vehicle`. Functionality provided by `bridge-sim` through this high-level API includes generation of finite element models based on a `Bridge` specification including mesh generation, traffic flow generation, settlement of piers and simulation of temperature effect. One of the generated models has been validated against sensor data collected from bridge 705 in Amsterdam, a plot comparing responses from linear simulation to static load tests (a truck parked on the bridge) is shown in Figure 2.

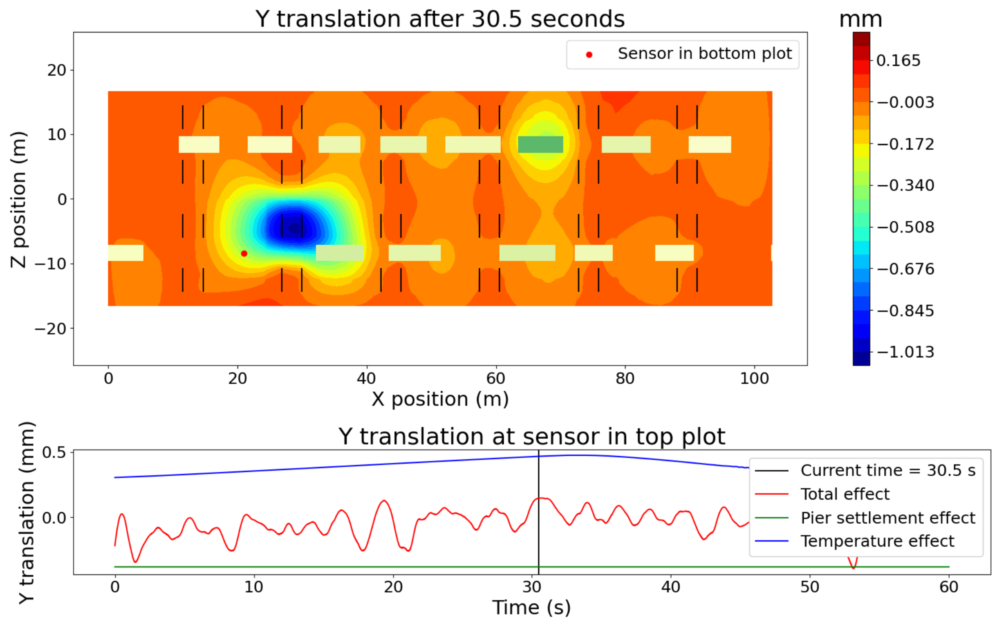

Figure 1: The top plot is a contour plot of vertical translation responses. The rectangles are vehicles on the bridge. One supporting pier has been settled by 1 mm. The bottom plot shows a time series of responses from a vertical translation sensor, position indicated in the top plot.

Problem Domain

The probability of a bridge to fail increases over time until it is no longer considered safe for use. Maintenance of a bridge is typically carried out when something goes wrong or according to a preventative maintenance schedule based on expert knowledge, neither approach making the best use of limited maintenance resources. Many bridges in Europe are reaching the end of their design life because of the post-war surge in bridge contruction, yet maintenance can extend the life of a bridge. Sensor data can be used to detect damage in real-time without the delay or cost of a real-time maintenance check.

One of the biggest problems for research into damage detection of concrete slab bridges is the collection of data (Worden & Manson, 2006). This is because bridges are expensive structures that we are simply not allowed to damage, except as they are being decommisioned but then traffic is no longer permitted on the bridge. To circumvent this issue a lot of research is based on numerical simulations. However models used in such research are typically not built for re-use, and so researchers must resort to creating their models from the low-level building blocks of nodes and forces. `bridge-sim` addresses the need for a high-level API for data collection from concrete slab bridge simulation.

Usage Example

The code below shows a very simple example that can be run with `bridge-sim`, to give an idea of the kind of API that is provided. This example uses a built-in `Bridge` instance and a simulation is run with a custom `Vehicle` placed on the bridge. A contour plot of vertical translation responses is then generated with `bridge-sim` library functions, which are built upon the popular Matplotlib library.

```python
import matplotlib.pyplot as plt
from bridge_sim import bridges, configs, fem, model, plot, vehicle
from bridge_sim.vehicle import Vehicle

new_vehicle = Vehicle(
    # Load intensity of each axle.
    kn=[5000, 4000, 4000, 5000, 7000],
    # Distance between each pair of axles.
    axle_distances=[2, 2, 2, 1],
    # Width of each axle, distance between point loads.
    axle_width=2.5,
    # Speed of the vehicle.
    kmph=20,
)

config = configs.opensees_default(bridges.bridge_example, shorten_paths=True)
point_loads = new_vehicle.to_point_load_pw(
    time=3.5, bridge=config.bridge, list=True)
responses = fem.responses(config, model.RT.YTranslation, point_loads)
plot.contour_responses(config, responses, point_loads)
plot.top_view_bridge(config, piers=True)
plt.tight_layout()
plt.show()
```

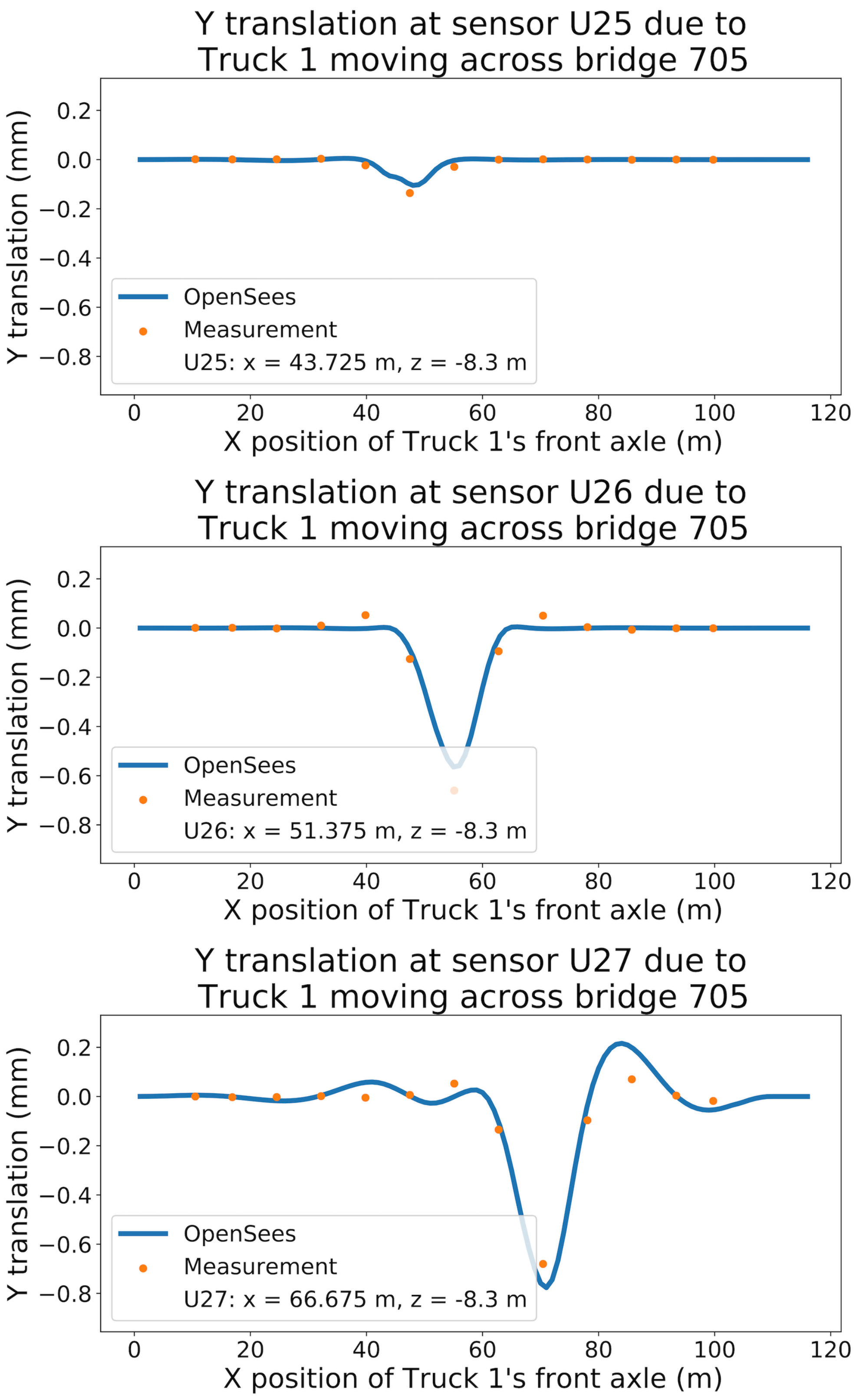

Figure 2: Comparison of vertical translation responses from linear simulation with `bridge-sim` and measurements collected in real life. The real bridge which is modeled and from which sensor measurements were taken is bridge 705 in Amsterdam. The x-axis in each plot shows the longitudinal position of the front axle of a truck parked on bridge 705. The y-axis shows the vertical translation from a sensor due to the truck's weight.

Acknowledgements

Support from TNO is acknowledged during the genesis of this project.

This software would not have been possible if not for a number of open source projects. In particular the contributors and authors of OpenSees (McKenna, 2011) deserve thanks.

References

McKenna, F. (2011). OpenSees: A framework for earthquake engineering simulation. *Computing in Science & Engineering*, *13*(4), 58–66.

Worden, K., & Manson, G. (2006). The application of machine learning to structural health monitoring. *Philosophical Transactions of the Royal Society A: Mathematical, Physical and Engineering Sciences*, *365*(1851), 515–537.