CFSRC COLLOQUIUM

Proceedings of the Cold-Formed Steel Research Consortium Colloquium
17-19 October, 2022 (cfsrc.org)

An open-source cold-formed steel connection test database to support future data models

Cristopher D. Moen[1], Baihan Chen[2]

Abstract

A data structure and physical test database are implemented for cold-formed steel connections. The database is publicly available on GitHub in a format that is both human-readable and machine-readable. The data structure is designed to accommodate connection tests with varying test variables including fastener type, quantity of fasteners in a specimen, and the number and type of plies. The database design is made compatible with open-source software tooling to support future data-driven model development. Strategies are discussed for encouraging general user adoption of open-source databases and software including education, community guidelines, and easy-to-use interfaces.

1. Introduction

Data on its own is data. Human-accessible data that can be downloaded, imported, and explored by anyone leads to new ideas and observations, especially if the data is also machine-readable [1], i.e., it has a data structure that can be accessed with software tooling. Strategies for making experimental data human- and machine-readable are explored in this paper with the design and construction of a cold-formed steel connection test database.

Driving a screw fastener through two plies of steel happens in a few seconds on a construction site, however creating a structural model that simulates these fasteners working together in a building currently takes much longer. As one connected ply moves relative to the other, the local connection stiffness changes as the screw tilts and tears through the plies. Fastener load-deformation response is complex and mostly available only as physical test data. There are powerful open-source tools and protocols for organizing and modeling with data that are continuously supported by a vast global network of volunteers. Why not try to use them?

For example, convenience functions for converting data from text files or Excel to a data structure, referred to herein as a `struct`, are available in most computer languages [2]. A `struct` can have the look and feel of a spreadsheet, for example, a `DataFrame` [3], but instead of a single value per field, they can hold any data format, from a scalar to array to an image.

[1]Cristopher D. Moen, RunToSolve, LLC cris.moen@runtosolve.com
[2]Baihan Chen, RunToSolve, LLC baihan.chen@runtosolve.com

Think about a `struct` in the context of a steel-to-steel screw-fastened connection test, where the steel ultimate stress f_u is used to predict connection strength. The connection response trends with the steel ultimate stress f_u, however strength prediction may correlate better to some other unknown model where the full stress-strain curves for the plies are utilized. The whole stress-strain curve from a test can be stored in a `struct`, i.e. `test.stress` and `test.strain`. Increased data fidelity, access, and organization enabled by data structures accommodates deeper data exploration.

And defining data models with a data structure is especially convenient these days. For example, Interpolations.jl provides fast, continuous multivariate interpolation of discrete data sets. This means that for a set of connection tests, with ply thicknesses t_1 and t_2 and f_y and fastener size and load P and relative ply displacement Δ, that a model can be created where $P_{model}, \Delta_{model} = f(t_1, t_2, f_y, f_u)$. Instead of writing down empirical equations that attempt to match the data, the model f, informed by a database, directly approximates connection load-deformation response, strength, and stiffness.

2. Database

The interactivity of this particular open-source cold-formed steel connection test database is defined by GitHub. Anyone can copy the database (Clone), post a question or bug (Issue), and submit data (Pull Request). The database exists as a JSON file which is a data interchange format that we all use every day on websites and mobile applications.

When a data set is submitted, it is reviewed and mapped to a data structure. The `struct` is then converted to JSON format, and written to the database JSON file.

For this database, all the information about a test specimen is carried in the data structure `Specimen` which is made up of `Source, Fastener, Ply, Test.`

```
struct Specimen
  source::Source
  fastener::Fastener
  ply::Ply
  test::Test
end

struct Source
  authors::Array{String}
  date::Date
  title::String
  bibtex::String
  units::Array{String}
end

struct Fastener
  type::Array{String}
  size::Array{Float64}
end

struct Ply
  type::Array{String}
  thickness::Array{Float64}
  elastic_modulus::Array::Array{Float64}
  yield_stress::Array{Float64}
  ultimate_stress::Array{Float64}
end

struct Test
  name::String
  loading::String
  force::Array{Float64}
  displacement::Array{Float64}
end
```

The data structure is designed with ease of modification and expansion in mind. The `struct` fields are written as arrays to hold multiple values, and `struct` fields can be added or removed. This means the database can hold a connection test where there was just one fastener or a test with 4 fasteners, or a test with 2 plies or 3 plies, or a data set with 1 author or 3 authors, or a test with gypsum ply and a steel ply, or a data set with some screw-fastened tests and some puddle weld tests.

Here are some examples of ways one might interact with the `Specimen` object. To access all of the researchers associated with a specimen, use `specimen.source.authors` or for just the first author `specimen.source.authors[1]`. Write `specimen.test.force` to obtain the actuator force vector or `specimen.ply.thickness[1]` to access the first measured ply thickness. A data set of many specimens can be represented as an array of `Specimen` objects.

This database has a unique web URL and it can be cited like any other publication [4]. To use the database, clone it, import the JSON file into a data structure with your favorite programming language, and then query the data structure. For example, find all connection tests in the database where the first ply thickness is less than or equal to 2.0 mm using the Julia language [5]:

```
using JSON3
json_string = read("fastener_connection_data.json",
String)
database = JSON3.read(json_string)

my_data = [database.test[i].ply.thickness[1]<=2.0u"mm"
for i in eachindex(database)]
```

3. Discussion

A database with a data structure is the jumping off point to data visualization, data-driven models, machine learning, and AI techniques.

Documentation that defines the data structure is essential for encouraging users. In this database, the `structs` are written with descriptive names, which helps ease the documentation burden. For example, `specimen.fastener.type` leads you to the data you are looking for with words.

The selection of a JSON file for this database was made because it is both human-readable and machine-readable, however large JSON datasets become cumbersome. Using an SQL database provides faster access to data sources. Storing large data sets in a compressed binary format is also common [6].

The data submission and review stage is important because it influences data model quality downstream [7]. Data review and mapping data to the data structure is time-consuming and requires trained volunteers. Providing motivation to publish test data directly in machine-readable format should be a high priority for any industry or institution.

The successful use of data requires a community of users [8]. It is important that both universities and industry invest in open-source database and software development and training if they want to make use of data-driven models [9].

Web-based user interfaces to a database are useful to engage users [10].

Units of measurement for each data type should be included in the data structure. Useful packages are available for automating unit conversion [11].

Open-source community rules and democratic workflows that encourage trust between maintainers and users are important [12].

It is challenging to balance data structure fidelity and usability. There are examples of complex data models where every parameter is documented [13], however the user is potentially burdened by this complexity. Thought and discussion and trial usage and iteration and testing are required to produce a useful open-source database.

4. Conclusion

An open-source cold-formed steel connection test database was designed and implemented. The database has a flexible data structure from which future data-informed prediction models can be built.

References

[1] Victor, Bret. "What can a technologist do about climate change?" (2015), [Online]. Available: `http://worrydream.com/ClimateChange/`.

[2] The Julia language. "Types." (2022), [Online]. Available: `https://docs.julialang.org/en/v1/manual/types/`.

[3] Jacob Quinn. "DataFrames.jl." (2022), [Online]. Available: `https://github.com/JuliaData/DataFrames.jl`.

[4] C. D. Moen and B. Chen, *FastenerConnectionData*, 2022. [Online]. Available: `https://github.com/runtosolve/FastenerConnectionData.git`.

[5] J. Bezanson, A. Edelman, S. Karpinski, and V. B. Shah, "Julia: A fresh approach to numerical computing," *SIAM review*, vol. 59, no. 1, pp. 65–98, 2017. [Online]. Available: `https://doi.org/10.1137/141000671`.

[6] The HDF Group. "Hierarchical data format version 5." (2000-2010), [Online]. Available: `http://www.hdfgroup.org/HDF5`.

[7] The National Science and Technology Council, "Desirable characteristics of data repositories for federally funded research," 2022. DOI: `https://doi.org/10.5479/10088/113528`.

[8] NumFocus, *NumFocus*, 2022. [Online]. Available: `https://numfocus.org/`.

[9] The Linux Foundation. "Participating in Open Source Communities." (2022), [Online]. Available: `https://www.linuxfoundation.org/resources/open-source-guides/participating-in-open-source-communities`.

[10] E. Dong, H. Du, and L. Gardner, "An interactive web-based dashboard to track covid-19 in real time," *The Lancet infectious diseases*, vol. 20, no. 5, pp. 533–534, 2020.

[11] CalTech Painter Lab, *Unitful.jl*, 2022. [Online]. Available: `https://github.com/PainterQubits/Unitful.jl`.

[12] V. DEVENYI, D. DI GIACOMO, C. DUSSUTOUR, B. KUDZMANAITE, and M. SHAIKH. "Guidelines for Sustainable Open Source Communities in the Public Sector." (2021), [Online]. Available: `https://joinup.ec.europa.eu/sites/default/files/inline-files/2021%20Updated%20Guidelines%20for%20creating%20sustainable%20OS%20communities_1.pdf`.

[13] S. J. Chalk, "Scidata: A data model and ontology for semantic representation of scientific data," *Journal of cheminformatics*, vol. 8, no. 1, pp. 1–24, 2016.