UNiT: A Universal Toolkit for Metallic Nanoindentation Data Analysis and Mechanical Properties Exploration

Advisors:

Professor Rodica Neamtu (Computer Science) and Professor Danielle Cote (Materials Science and Engineering)

By:

Aaron Krueger (Computer Science) and Eric Schmid (Computer Science)

Working collaboratively with:

Katelyn Barron (Mechanical Engineering)

Bryer Sousa (Materials Science and Engineering)

A Major Qualifying Project

WORCESTER POLYTECHNIC INSTITUTE

August 2021 – March 2022

ABSTRACT

With vast amounts of data created by nanoindentation testing, materials science researchers often find themselves struggling to analyze and utilize said data efficiently and effectively. As a result, the goal of this project was to develop the Universal Nanoindentation Toolkit (UNiT). The toolkit implemented methods to analyze nanoindentation data in ways that were not previously available to materials scientists, providing improved accessibility for these methods and allowing for more significant analysis of nanoindentation data. In addition, the user interface was developed and refined through user testing, and the implementation of each method was validated against other materials science tests to prove its accuracy.

ACKNOWLEDGEMENTS

The team would like to thank Professors Neamtu and Cote, as their dedication to the project helped it grow and thrive. Also integral to the project's success was Bryer Sousa, whose materials science knowledge significantly improved the team's ability to work efficiently. Lastly, feedback from those participating in user testing was extremely helpful in identifying areas of success and areas for improvement for the toolkit.

TABLE OF CONTENTS

TABLE OF FIGURES

TABLE OF EQUATIONS

TABLE OF TABLES

INTRODUCTION

Large datasets have become increasingly prevalent in modern academia and industry. As a result, materials scientists often find themselves working with tools that provide massive amounts of information; however, analyzing this data often proves time-consuming and difficult from both a resource and computation perspective. Specifically, nanoindentation – the method of continually pressing a rigid indenter into some material – can quickly generate tens of thousands of data points with several features in a single test. More closely examining materials science processes like nanoindentation shows that precision and efficiency are paramount. Even so, working with datasets produced by nanoindentation often proves unwieldy and inefficient when done manually.

Some researchers have set out to tackle this efficiency issue; Campbell et al. (2019) created the Nanoindentation General Evaluation Tool (NIGET) to address a lack of access to software that fits their needs for nanoindentation evaluation outside of the default calculations provided by instrument manufacturers. For example, the authors describe a lack of uncertainty analysis as a significant shortcoming of this default instrument software and generally inadequate adaptability for different datasets (Campbell et al., 2019). As a result, the authors created NIGET to perform various operations on nanoindentation data, such as the Oliver-Pharr method, pop-in detection, and elastic and plastic work calculations (Campbell et al., 2019).

While NIGET covers many general evaluation methods, materials scientists also struggle to utilize modern analysis methods due to a lack of awareness and availability. Also, many materials scientists do not have the technical background required to understand and implement the methods in total. Providing these more detailed analysis tools would pave the way for researchers to glean more information from their data, such as calculating hardness in the limit of infinite depth and any associated properties described in Chen Bull (2009) and Haušild (2021).

This project aims to create the Universal Nanoindentation Toolkit (UNiT), which addresses the lack of availability for calculating specific metrics and measurements about nanoindentation data. In doing so, UNiT will provide a method for materials scientists to analyze their data more efficiently and accurately than with default tools provided by nanoindentation instruments. In addition, UNiT will provide experimental methods from more modern research not available through other frameworks like NIGET. Finally, UNiT will be tested extensively through several methods, such as tensile testing, stress-strain evaluation, and Vickers hardness testing. In conclusion, the combination of materials science testing and software development will refine UNiT into a fully functional analysis tool for nanoindentation data.

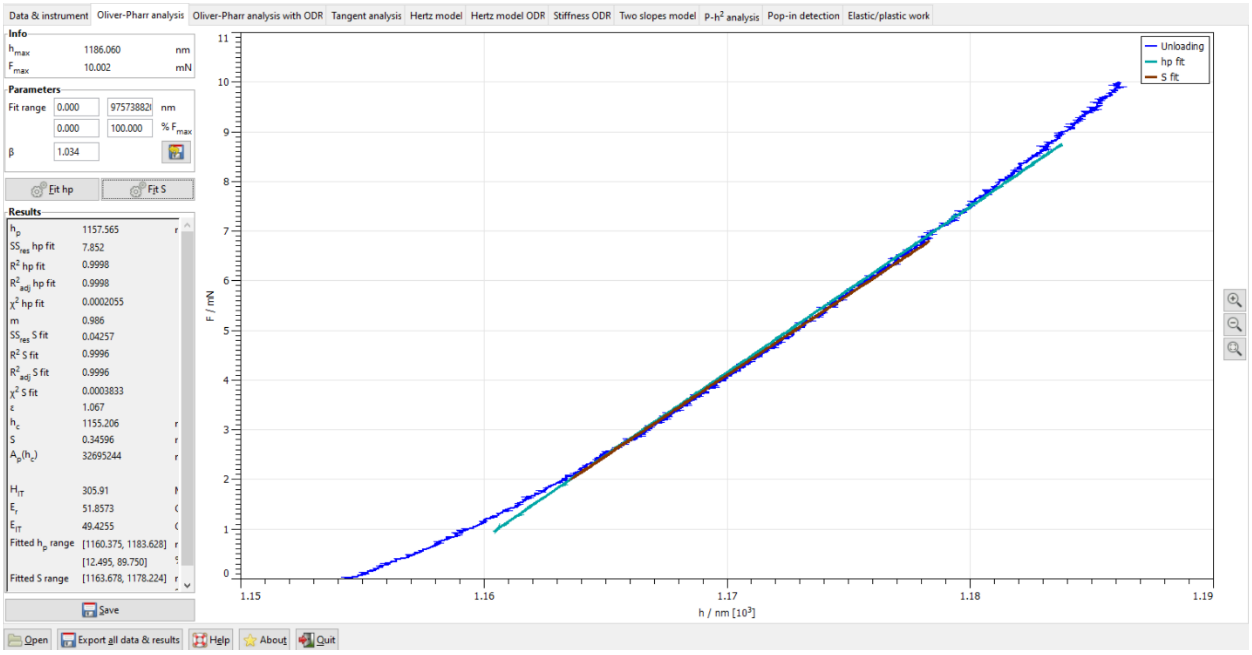
BACKGROUND

Nanoindentation is a materials science and engineering characterization technique used to assess the mechanical properties of materials by repeatedly pressing an indenter into the material and subsequently analyzing the load-displacement data generated by this procedure. To process the large amount of data created through this technique, commercially available nanoindentation equipment provides general evaluation suites that allow the user to calculate basic metrics of the data. However, this software often does not provide the user with methods to perform advanced evaluation and analysis on the load-displacement data. This presents a lack of accessibility for many of the calculations required to evaluate and analyze nanoindentation data thoroughly. Attempting such evaluation by hand can be an inefficient and error-prone process.

The integration of software solutions in materials science and engineering has a strong background in recent years. Materials science processes often generate significantly large datasets prime for computer-automated evaluation and analysis. As a result, many materials science and engineering research groups have adopted data-driven processes and improved data discoverability (Plante et al., 2021). For example, organizations such as the Open Quantum Materials Database and several universities with well-developed materials science curricula seek to make datasets more widely available and further increase researchers' abilities to process such data by publishing it online and improving its accessibility (Hill et al., 2016). As a result, some researchers and organizations have created software solutions such as the Nanoindentation General Evaluation Toolkit and the PopIn Toolbox to address this issue.

The Nanoindentation General Evaluation Toolkit

The Nanoindentation General Evaluation Toolkit, or NIGET, was created to allow researchers to utilize evaluation methods not present in commercial nanoindentation instruments (Campbell et al., 2019). In creating this application, Campbell et al. gave researchers more room to implement additional methods for evaluation and easily compare results from different methods through a simple interface. The tool is more modular than commercial software, with users being given multiple separate tools to evaluate datasets. NIGET offers several calculations common to evaluating load-displacement data, including Oliver-Pharr analysis, pop-in detection, elastic and plastic work calculation, and more **(Figure 1)**. NIGET developers also utilized a new method of fitting load-displacement curves known as orthogonal distance regression (ODR). They found that this method more closely fits the curve than the standard method of ordinary least squares (OLS). Lastly, NIGET offers a method to calculate uncertainties of the results of these methods, such as through Monte Carlo and Gaussian propagation methods. Overall, NIGET provides many valuable general evaluation tools; even so, there are several newer methods of detailed analysis that materials scientists and engineers would benefit from utilizing within an evaluation suite.



Info

h_{max}	1186.060	nm
F_{max}	10.002	mN

Parameters

Fit range	0.000	975738826	nm
	0.000	100.000	% F_{max}
β	1.034		

Results

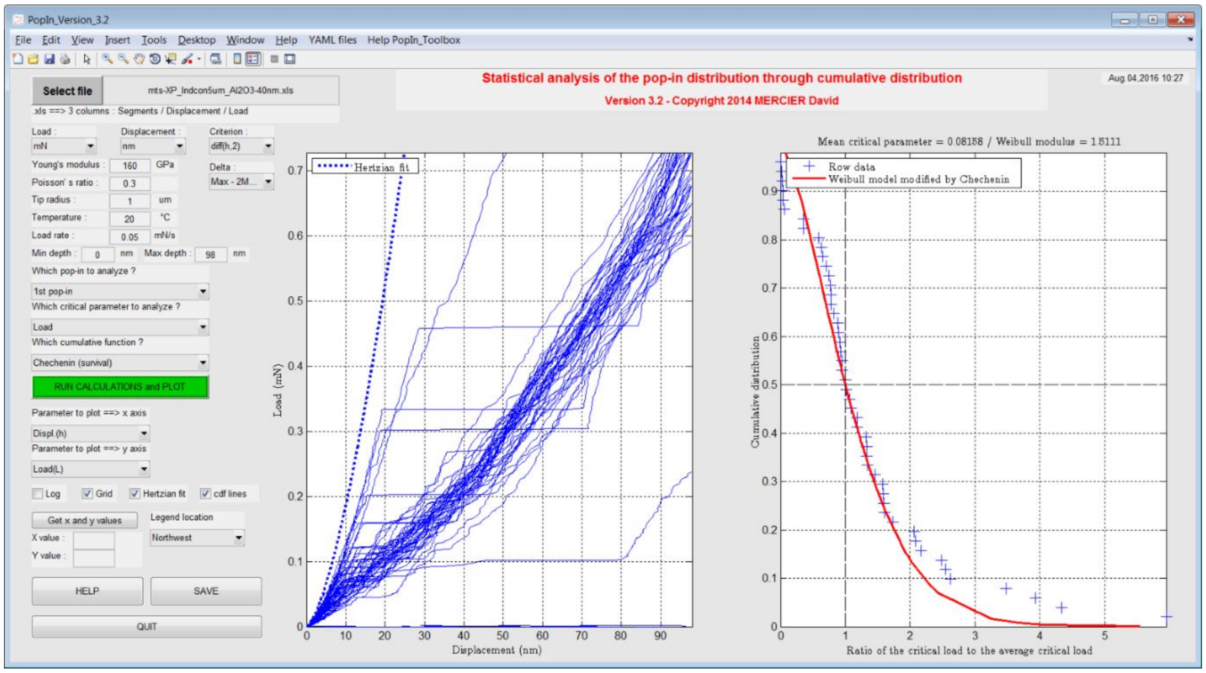
h_p	1157.565	r
SS_{res} hp fit	7.852	
R^2 hp fit	0.9998	
R^2_{adj} hp fit	0.9998	
χ^2 hp fit	0.0002055	
m	0.986	
SS_{res} S fit	0.04257	
R^2 S fit	0.9996	
R^2_{adj} S fit	0.9996	
χ^2 S fit	0.0003833	
ε	1.067	
h_c	1155.206	r
S	0.34596	r
$A_p(h_c)$	32695244	r
H_{IT}	305.91	N
E_r	51.8573	(
E_{IT}	49.4255	(
Fitted h_p range	[1160.375, 1183.628]	r
	[12.495, 89.750]	9
Fitted S range	[1163.678, 1178.224]	r

Figure 1 – Oliver-Pharr analysis using NIGET

As for usability features, users are given the option to upload load-displacement curve data and save their analysis data in simple text files. However, it is essential to note that NIGET's data upload options are rather strict; files must fit a predefined plain-text format and cannot contain infinite or "Not a Number" (NaN) values known to cause errors when plotting data or performing calculations. This is an issue, as nanoindentation data is often output in .csv or .xlsx format and a relatively significant portion of this data contains infinite or NaN values in some rows. For example, this can occur in data collected before the indenter reaches a depth of 30 nm, as inconsistencies in the material's surface may compromise output. Furthermore, some datasets produce upwards of 10,000 rows for a single test, and many materials scientists may not have the ability to fix or remove rows with prohibited values quickly. As for saving data, if the user exits before saving, their results will not be saved. With such issues surrounding uploading and saving data, one would certainly recognize that a native dataset cleaning suite and non-volatile data storage would benefit the user greatly.

The PopIn Toolbox

The PopIn Toolbox was created using MATLAB to detect an event known as a "pop-in" in load-displacement data. A pop-in is essentially a phenomenon where the load-displacement curve suddenly plateaus or drops vertically (Mercier, 2018). This generally occurs due to cracks forming in the material being indented, phase transformation, or rupture of a brittle film on the substrate (Mercier, 2018). The PopIn Toolbox allows users to specify various parameters relevant to pop-ins, such as which pop-in to analyze, the parameter to analyze, and the size of the pop-in. It also provides the user with visualizations of data related to where pop-ins occur **(Figure 2)**.


PopIn_Version_3.2
File Edit View Insert Tools Desktop Window Help YAML files Help PopIn_Toolbox
Select file
RUN CALCULATIONS and PLOT
Get x and y values
HELP
SAVE
QUIT
Hertzian fit
Load (mN)
Displacement (nm)
Row data
Weibull model modified by Chechenin
Cumulative distribution
Ratio of the critical load to the average critical load

Statistical analysis of the pop-in distribution through cumulative distribution
Version 3.2 - Copyright 2014 MERCIER David
Aug.04.2016 10:27

mts-XP_Indcon5um_Al2O3-40nm.xls
.xls ==> 3 columns : Segments / Displacement / Load

Load : mN
Displacement : nm
Criterion : diff(h,2)
Young's modulus : 160 GPa
Delta : Max - 2M...
Poisson' s ratio : 0.3
Tip radius : 1 um
Temperature : 20 °C
Load rate : 0.05 mN/s
Min depth : 0 nm Max depth : 98 nm
Which pop-in to analyze ? 1st pop-in
Which critical parameter to analyze ? Load
Which cumulative function ? Chechenin (survival)
Parameter to plot ==> x axis : Displ.(h)
Parameter to plot ==> y axis : Load(L)
☐ Log ☑ Grid ☑ Hertzian fit ☑ cdf lines
Legend location : Northwest
X value :
Y value :

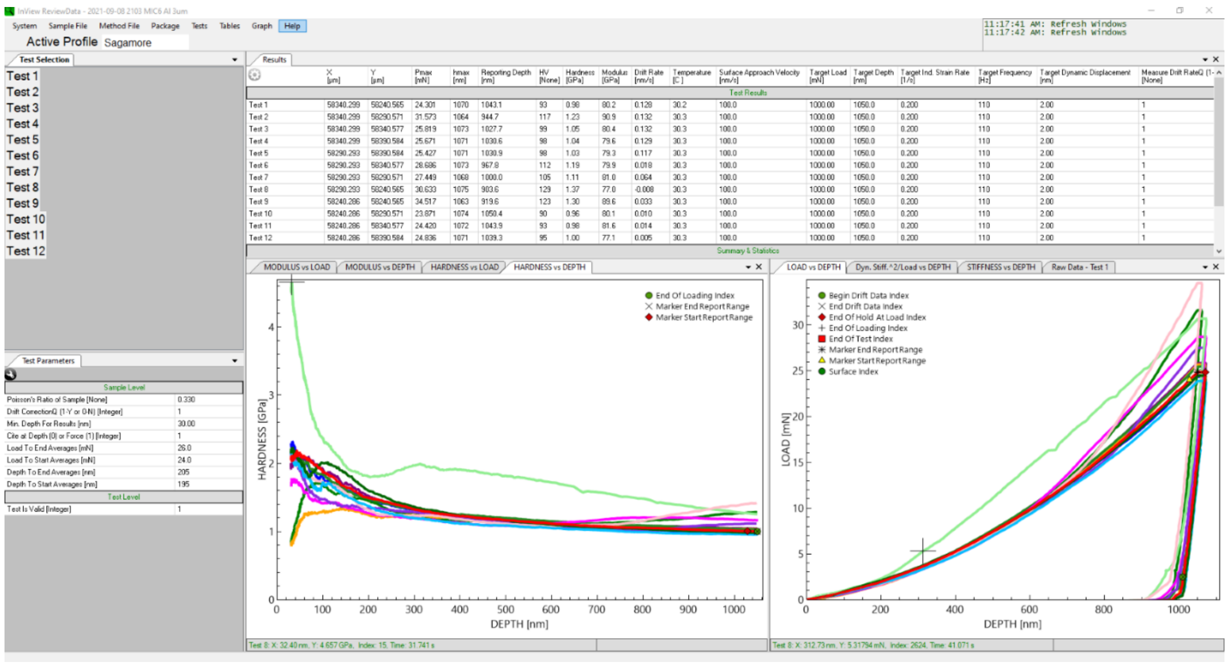
Mean critical parameter = 0.08158 / Weibull modulus = 1.5111

Figure 2 – The PopIn Toolbox

NIGET also offers a pop-in detection tool, allowing users to specify the size of pop-ins to detect in a load-displacement curve. The advantage of the PopIn Toolbox is that it allows much more in-depth analysis than NIGET's tool; NIGET only offers a few parameters that essentially just allow users to visualize pop-ins in their data, whereas the PopIn Toolbox allows users to select several parameters and visualization aids when detecting pop-ins. On the other hand, NIGET offers a centralized toolkit that allows users to detect pop-ins while also calculating many other metrics about the data. A tool that allows users to perform in-depth analysis and utilize several evaluation methods would prove invaluable to many materials scientists.

Nanoindentation Equipment Software

Nanoindentation equipment generally provides its users with software to visualize and perform calculations on nanoindentation data. One example is the InView Review Program provided with the NanoBlitz 3D nanoindenter (Nanomechanics Inc., 2018). The software allows the user to configure various input settings for a specific test, such as a target depth, load, and strain rate. After running the test with these inputs, the machine will provide helpful visualizations of different aspects of the data, such as scatter plots of load and depth data, histograms demonstrating values of different data features, and property maps of specific data features **(Figure 3)**. The software will also allow users to access individual indentation results and summary statistics for the test. These options enable users to perform their calculations on individual results manually or quickly analyze the data through summaries.



```
11:17:41 AM: Refresh Windows
11:17:42 AM: Refresh Windows
```

	X [µm]	Y [µm]	Pmax [mN]	hmax [nm]	Reporting Depth [nm]	HV [None]	Hardness [GPa]	Modulus [GPa]	Drift Rate [nm/s]	Temperature [C]
Test Results										
Test 1	58340.299	58240.565	24.301	1070	1043.1	93	0.98	80.2	0.128	30.2
Test 2	58340.299	58290.571	31.573	1064	944.7	117	1.23	90.9	0.132	30.3
Test 3	58340.299	58340.577	25.819	1073	1027.7	99	1.05	80.4	0.132	30.3
Test 4	58340.299	58390.584	25.671	1071	1030.6	98	1.04	79.6	0.129	30.3
Test 5	58290.293	58390.584	25.427	1071	1030.9	98	1.03	79.3	0.117	30.3
Test 6	58290.293	58340.577	28.686	1073	967.8	112	1.19	79.9	0.018	30.3
Test 7	58290.293	58290.571	27.449	1068	1000.0	105	1.11	81.0	0.064	30.3
Test 8	58290.293	58240.565	30.633	1075	903.6	129	1.37	77.0	-0.008	30.3
Test 9	58240.286	58240.565	34.517	1063	919.6	123	1.30	89.6	0.033	30.3
Test 10	58240.286	58290.571	23.871	1074	1050.4	90	0.96	80.1	0.010	30.3
Test 11	58240.286	58340.577	24.420	1072	1043.9	93	0.98	81.6	0.014	30.3
Test 12	58240.286	58390.584	24.836	1071	1039.3	95	1.00	77.1	0.005	30.3
Summary & Statistics										

	Surface Approach Velocity [nm/s]	Target Load [mN]	Target Depth [nm]	Target Ind. Strain Rate [1/s]	Target Frequency [Hz]	Target Dynamic Displacement [nm]	Measure Drift RateQ (1- [None]
Test Results							
Test 1	100.0	1000.00	1050.0	0.200	110	2.00	1
Test 2	100.0	1000.00	1050.0	0.200	110	2.00	1
Test 3	100.0	1000.00	1050.0	0.200	110	2.00	1
Test 4	100.0	1000.00	1050.0	0.200	110	2.00	1
Test 5	100.0	1000.00	1050.0	0.200	110	2.00	1
Test 6	100.0	1000.00	1050.0	0.200	110	2.00	1
Test 7	100.0	1000.00	1050.0	0.200	110	2.00	1
Test 8	100.0	1000.00	1050.0	0.200	110	2.00	1
Test 9	100.0	1000.00	1050.0	0.200	110	2.00	1
Test 10	100.0	1000.00	1050.0	0.200	110	2.00	1
Test 11	100.0	1000.00	1050.0	0.200	110	2.00	1
Test 12	100.0	1000.00	1050.0	0.200	110	2.00	1
Summary & Statistics							

Sample Level	
Poisson's Ratio of Sample [None]	0.330
Drift CorrectionQ (1-Y or 0-N) [Integer]	1
Min. Depth For Results [nm]	30.00
Cite at Depth (0) or Force (1) [Integer]	1
Load To End Averages [mN]	26.0
Load To Start Averages [mN]	24.0
Depth To End Averages [nm]	205
Depth To Start Averages [nm]	195
Test Level	
Test Is Valid [Integer]	1

Test 8: X: 32.40 nm, Y: 4.657 GPa, Index: 15, Time: 31.741 s

Test 8: X: 312.73 nm, Y: 5.31794 mN, Index: 2624, Time: 41.071 s

Figure 3 – Analysis of various curves and data using the InView evaluation software

While essential evaluations are available through summary statistics, most nanoindentation equipment software creates solid visualizations. Even though these visualizations can be helpful, materials scientists and engineers may struggle to use more detailed analysis methods on the large datasets provided by the software. Nanoindentation software does not provide analysis methods like pop-in detection, elastic and plastic work calculation, and uncertainty propagation. This represents an excellent opportunity for other software to provide researchers with access to more advanced analysis methods like those described above.

UNiT: The Universal Nanoindentation Toolkit

To address the shortcomings described above, our goal is to develop a software application that will provide materials scientists with the tools necessary to perform a detailed analysis of nanoindentation data. The application, known as the Universal Nanoindentation Toolkit (UNiT), will implement methods such as Nix-Gao's calculation of hardness at the limit of infinite depth, elastic and plastic work based on load-displacement curves, and more. These algorithms generally are not provided by software that comes with commercially available nanoindentation equipment; furthermore, tools like NIGET and the PopIn Toolbox do not enable users to perform additional computations that are not already available in commercially available software such as the InView Review Program. UNiT will also have improved user-friendliness compared to the tools described above, with the target audience consisting mainly of materials scientists who do not have strong technical backgrounds in computational analysis. As such, UNiT will have a more welcoming user interface, allow users to save progress and results more easily than tools like NIGET, and perform several relevant forms of advanced analysis without needing to access separate software.

METHODOLOGY

This section will cover the tools implemented in the Universal Nanoindentation Toolkit (UNiT) and many design decisions made throughout the project. This includes specific implementation details of each algorithm, the technology used in the project, and the design of the user interface.

Technology Stack

UNiT was implemented using a Python backend, including libraries such as SciPy (Scipy, n.d.) and Scikit-Learn (Scikit-Learn, n.d.). The SciPy library provides high-level access to valuable functions implemented in low-level, fast languages like C and FORTRAN. UNiT utilized SciPy functionality in curve fitting and integration, both of which were valuable for much of its functionality (Scipy, n.d.). Scikit-learn allows users to efficiently perform machine learning-related functionality like linear regression, which also proved valuable for the various algorithms implemented in UNiT (Scikit-Learn, n.d.). The user interface was created using TypeScript, React, and Material UI. The user interface and backend were integrated using a REST API implemented through Flask. While developing the toolkit, the team attempted to optimize its modularity so future users could easily add new algorithms. For example, the user interface was designed to allow for any number of new algorithms to be added to UNiT and for algorithms to provide graphical output alongside numerical output. A single class handles many initial steps for algorithms, such as data preprocessing, meaning users simply need to implement their algorithms and hook them up to the main class before integrating them with the user interface. Modularity is essential here, as it provides a method by which future developers and users can quickly improve upon and update UNiT with new analysis methods. The toolkit was also designed with the help of materials scientists, allowing the team to improve its usability by targeting more specific user groups. Usability was further improved by UNiT's ability to be hosted in the browser, allowing users to bypass a potentially complex installation process and simply open a webpage and begin working immediately.

Algorithms

With the help of materials scientists experienced in nanoindentation research, the team identified three main algorithms to implement in UNiT. These algorithms were chosen because they have little software available to calculate them and would allow materials scientists to access complex methods described in historically significant nanoindentation research papers. The first algorithm analyzes indentation size effects via strain gradient plasticity in crystalline materials to determine Vickers hardness of a material based on William Nix and Huajian Gao's 1998 paper (Nix & Gao, 1998). This algorithm was further refined through a secondary method to determine Vickers hardness described in (Haušild, 2021). Both steps are completed by implementing calculations described in each paper, mainly linear regression and algebra. The second algorithm determines stress and strain values for a given material by solving for variables through several complex formulas described in (Dao et al., 2001). The third and final algorithm calculates the ratio between the hardness and reduced Young's modulus of a material as described in (Bull & Chen, 2009). Much of the functionality of this algorithm relies on straightforward calculations described in the literature while also relying on manipulating the data to calculate the integrals of various data curves. Materials scientists and engineers would benefit from having access to computational software that can evaluate the algorithms described here, as the algorithms are often complex enough that manually performing calculations would be complicated, inefficient, and error-prone. UNiT will provide access to more modern and less accessible analysis techniques for several material properties, avoiding algorithms already implemented by most modern nanoindentation equipment. This will allow UNiT to maintain a codebase free of extraneous functionality, especially that of already standard or dated analysis methods.

Datasets

Nanoindentation datasets are procured using nanoindentation equipment like the KLA iMicro Nanoindenter to run nanoindentation tests. This device can take thousands of accurate measurements throughout a single test. Test data includes information on different material features such as the material's hardness, Young's modulus, and stiffness, and information about the indenter itself such as its depth and the force it exerts on the material. Frequently, nanoindentation researchers perform several tests on a material. When compounded with the thousands of rows of data produced by a single test, this process results in large datasets which are often stored in Excel files. Nanoindentation datasets can reach a magnitude of as much as 50 megabytes of data per dataset, with each test containing about 10,000 rows of data. Each row represents a single feature measurement, and each feature is stored in a single column. Features are generally measured with floating-point numbers, and units are described by the user running the test and output as a sub-header in each column.

A feature can have empty values in some cases as the nanoindenter has not begun the test properly yet. For example, if the indenter has yet to gather enough readings on the material, then the dataset will include depth values but not hardness values for that moment in time. Again, this magnitude of data proves the importance of software in performing computations. UNiT can read and perform calculations on more than one test sheet at a time and simply requires the user to select a valid Excel document containing nanoindentation data to begin performing calculations.

Algorithms: General Structure & Modularity

Each algorithm followed a similar structure intended to allow for improved code modularity. When implementing an algorithm, a Python class would be created in a specific folder that held all other algorithms. The constructor of this class would take any user input needed for the algorithm to function, including nanoindentation data. This data is generally stored in Pandas DataFrame objects, which are essentially two-dimensional objects used to store any amount of data (Pandas Documentation, n.d.). Each class has a function that roughly correlates to a calculation from the relevant paper for the algorithm and a method to run all calculations and provide whatever necessary output. To further improve modularity, UNiT provides a main algorithm running class. This class handles all data preprocessing, storing features from the dataset as Pandas DataFrames in a single object. This way, users can easily pass in whatever data they need for their algorithm to run. In terms of modularity, when an algorithm is implemented, the user simply needs to create a method to instantiate the algorithm's class and call whatever methods they need to gather output. If the algorithm implements the primary method to perform calculations, this is as straightforward as creating the algorithm object, calling the primary calculation method, and returning any output.

Algorithm 1: Determining Vickers Hardness and h*

A method described in William Nix and Huajian Gao's 1998 paper "Indentation size effects in crystalline materials" describes how to relate a material's hardness and depth through the following formula:

$$\frac{H}{H_0} = \sqrt{1 + \frac{h^*}{h}}$$

Equation 1 – The original Nix Gao method of determining Vicker's Hardness

This method is commonly referred to as the "Nix-Gao" method in materials science literature. Vickers hardness, which is used to calculate a material's hardness in the limit of infinite depth (or actual hardness), is described by the variable H_0, and h* is a characteristic length that depends on the shape of the indenter, the shear modulus, and H_0. UNiT is capable of determining both H_0 and h* given any nanoindentation dataset containing hardness, H, and depth, h, readings and simply requires the user to input values for a material's shear modulus, μ, and Burgers vector, b. The user is also given the option to input a value θ that represents the angle between the surface of the indenter and the plane of the surface. θ is a necessary value to calculate the contact radius of the indenter and the material **(Equation 2)**. UNiT also provides a default of $\theta = 65.03°$, the angle for common Berkovich tips.

$$a = \frac{h}{\tan(\theta)}$$

Equation 2 – The calculation of a, contact radius

Lastly, a constant α is described as a default of 0.5 in the original Nix-Gao paper; this value is offered at this default, but users can also input their measurement for it. Values for h and H are stored in Pandas DataFrames, and all other values are simply stored as floats.

UNiT starts by reading h (depth) and H (hardness) values from the specified nanoindentation test to evaluate the Nix-Gao formula. First, both h and H are trimmed to the point where the h vs. H curve is only decreasing as described in (Wright et al., 2013). More specifically, this means that the valid h and H values are those occurring at a peak H after the point where $h = 200$ nm **(Figure 4)**. If this alteration were not performed, then the Nix-Gao method would consistently overestimate the value of H_0, leading to a loss of accuracy in all of the method's calculations.

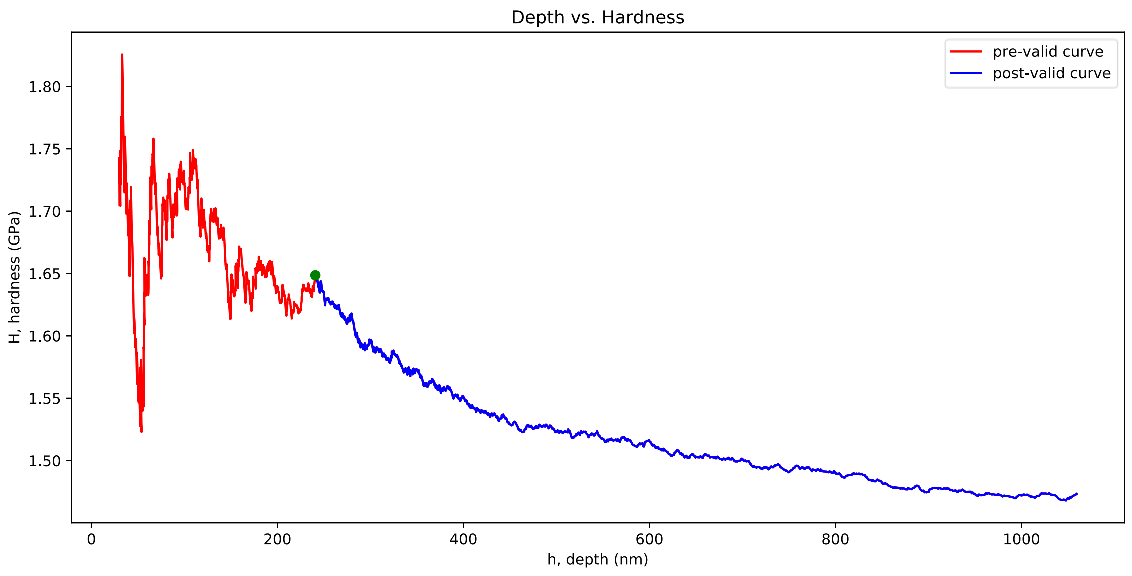


Figure 4 - The plot of H vs. h, where the blue curve represents the "valid" values of h and H used in subsequent calculations

UNiT then performs linear regression on the graph of *1/h* as a function of H^2 to determine H_0 as described in (Wright et al., 2013) **(Figure 5)**. The linear regression model is calculated using the Scikit-learn library's `LinearRegression` class, which provides a fit and subsequent predictions for $1/h$ and H^2. Plotting $1/h$ values, H^2 values, and the linear regression fit was done using the Matplotlib library. This library provides a comprehensive interface for creating data charts and was used throughout development to visualize data (Matplotlib, n.d.).

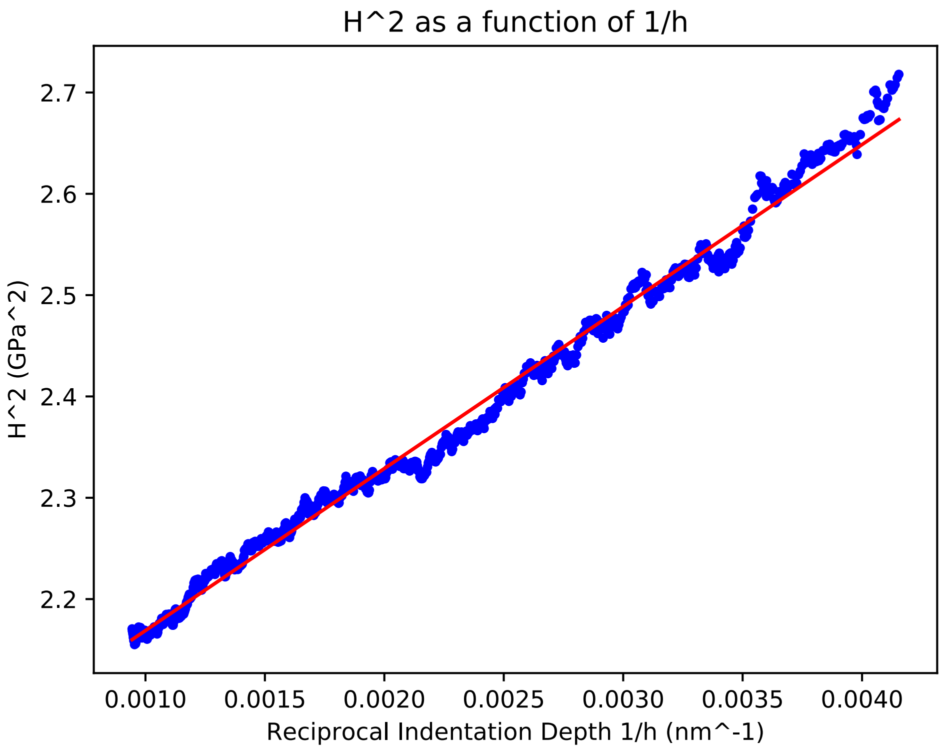


Figure 5 - H^2 as a function of 1/h, used to calculate H_0 as the square root of the y-intercept

After obtaining *H_0* and *h^**, UNiT can calculate *L_s*, or the mean spacing between statistically stored dislocations using ρ_s, the density of statistically stored dislocations **(Equations 3 and 4)**.

$$\rho_s = \left(\frac{H_0}{3\sqrt{3\alpha\mu b}}\right)^2$$

Equation 3 - The calculation of ρ_s, the density of statistically stored dislocations

$$L_s \cong \sqrt{\frac{1}{\rho_s}}$$

Equation 4 - The calculation of L_s, the mean spacing between statistically stored dislocations

UNiT also performs further calculations to determine the total dislocation density of the indentation, ρ_T, which requires the calculation of geometrically necessary dislocations, ρ_G **(Equations 5 and 6)**.

$$\rho_G = \frac{3h}{2ba^2}$$

Equation 5 - The calculation of ρ_G, the geometrically necessary dislocations

$$\rho_T = \rho_G + \rho_s$$

Equation 6 - The calculation of ρ_T, the total dislocation density of the indentation

At this point, UNiT has calculated the output of the formulas the team deemed necessary from the 1997 Nix Gao paper. Since the calculations are relatively straightforward when provided with user input, most of the implementation simply required standard algebra and little use of notable Python libraries. A secondary method also exists to determine hardness in the limit of infinite depth, as described in “On the breakdown of the Nix-Gao model for indentation size effect” **(Equation 7)** (Haušild, 2021).

$$H = H_0\sqrt{1 + \frac{h^*}{h}\left(1 + re^{-\frac{h}{h_1}}\right)^{-3}}$$

Equation 7 - The calculation of Vickers hardness using Hausild’s method

The variables *r* and *h₁* are described as fitting parameters, meaning UNiT must solve for them based on other available data. In this case, *H* is the dependent variable, and *h*, H_0, and *h** are independent variables. Using SciPy’s `curve_fit` function, UNiT can determine the values of *r* and h_1 and subsequently use their values in its calculation of hardness in the limit of infinite depth **(Figure 6)**. `curve_fit` works by using non-linear least squares to fit some function to data and finding the values for any fitting parameters used in said function (Scipy Optimize Curve_fit, n.d.). In this case, the fitted function is that in **Equation 7**, and `curve_fit` uses h, h^*, H_0, and H values to determine the values for r and h_1.

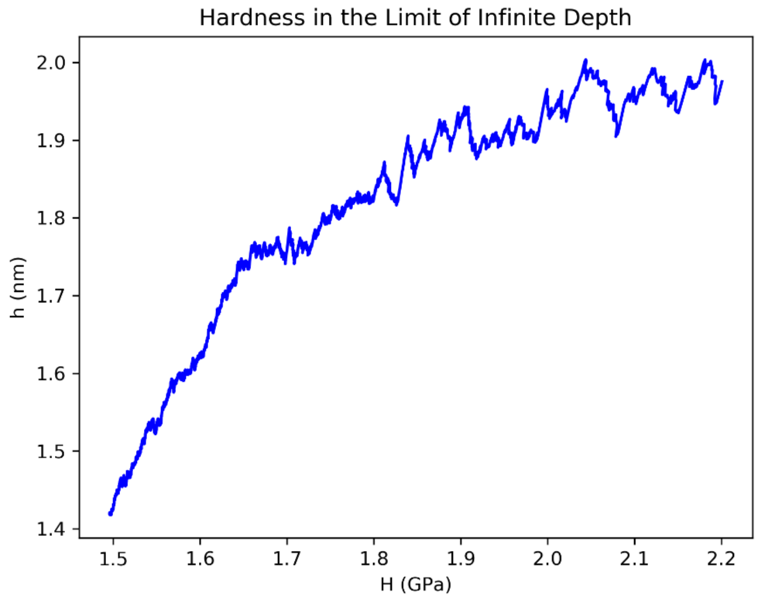


Figure 6 - Hardness in the limit of infinite depth, or actual hardness, as calculated using Hausild’s method

Algorithm 2: Evaluating a Stress-Strain Curve

Dao et al.'s 2001 paper "Computational modeling of the forward and reverse problems in instrumented sharp indentation" describes a forward and reverse method by which users can calculate stress and strain values given nanoindentation data (Dao et al., 2001). While both methods are valuable, the team decided to use the reverse analysis method as it proved more applicable to UNiT's domain and required less user input to operate than the forward analysis method. UNiT takes h (depth), P (load), E (Young's modulus), and ν (Poisson's ratio for the material) as user input. P, h, and E are stored in Pandas DataFrames, and ν is stored as a floating-point number. P and h form a load-depth curve, which, in nanoindentation applications, describes the general path of the indenter during its loading and unloading phases **(Figure 7)**.

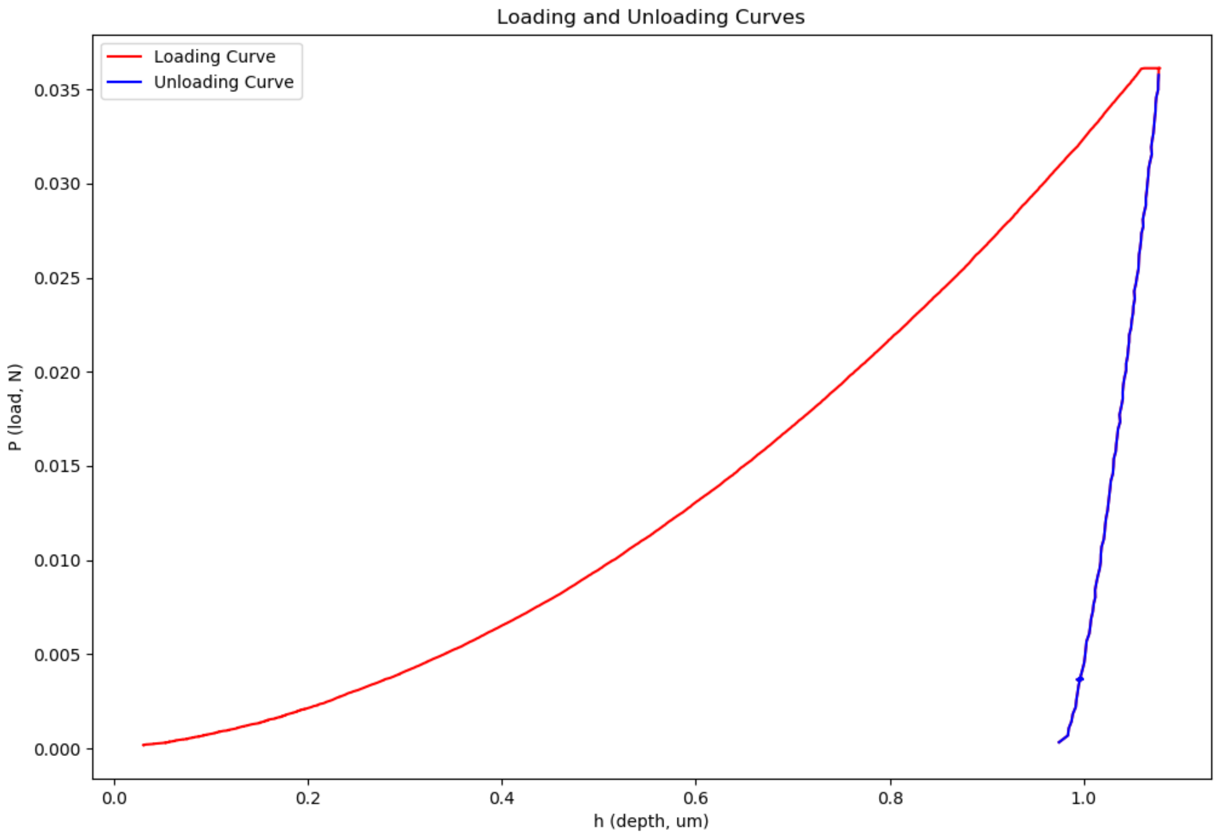


Figure 7 - The load-unload curve created by h and P values calculated during a nanoindentation test

Given this input, UNiT can calculate several other parameters for the data, including stress and strain values and computations specified in Dao et al.'s paper required for analysis. To achieve this, UNiT once again utilizes curve fitting functions like the SciPy library's `curve_fit` and `fsolve` functions, as several of the equations cannot be solved without fitting for specific parameters. UNiT starts by using the unloading curve created by h and P to solve for the residual depth after indentation, h_r, when $P = 0$. Specifically, h_r is solved by fitting h and P using the Numpy library's Polynomial `fit` function. This function fits an n-degree polynomial to x and y data, which, in this case, is used to solve for where $P = 0$. UNiT then solves for the loading curvature C based on max values of h and P **(Equation 8)**.

$$P_m = C * h_m^2$$

Equation 8 - Kick's Law for determining loading curvature

Next, UNiT calculates E^*, a value representing the reduced Young's modulus of the material **(Equation 9)**. UNiT provides default values for ν_i and E_i based on a diamond indentation tip, a standard tip used during nanoindentation.

$$E^* = \left(\frac{1-\nu^2}{E} + \frac{1-\nu_i^2}{E_i}\right)^{-1}$$

*Equation 9 - The calculation of E**

Using E^* and h_f, UNiT determines the average pressure p_{ave} before further calculating stress, σ, and strain, ϵ, data values. Specifically, UNiT calculates values for $\sigma_{0.082}$ and $\sigma_{0.033}$, which are described in Π_7 **(Equation 10)** and Π_1 **(Equation 11)** respectively in Dao et al.'s paper (Dao et al., 2001). In this case, UNiT uses `fsolve` to solve for both $\sigma_{0.082}$ and $\sigma_{0.033}$. `fsolve` simply calculates the x-intercept of a curve (Scipy Optimize Fsolve, n.d.), which is the value found when solving for $\sigma_{0.082}$ and $\sigma_{0.033}$ and setting the formula to 0. Since all other variables are already known in Π_7 and Π_1, `fsolve` is a great candidate for determining $\sigma_{0.082}$ and $\sigma_{0.033}$.

$$\frac{p_{ave}}{\sigma_{0.082}} = -15.4944\left(\frac{\sigma_{0.082}^2}{E^{*2}}\right) - 15.1699\left(\frac{\sigma_{0.082}}{E^*}\right) + 2.7497$$

Equation 10 - Dao et al.'s Π_7, used to solve for $\sigma_{0.082}$

$$\frac{C}{\sigma_{0.033}} = -1.131\left(\ln\left(\frac{E^*}{\sigma_{0.033}}\right)\right)^3 + 13.635\left(\ln\left(\frac{E^*}{\sigma_{0.033}}\right)\right)^2 - 30.594\left(\ln\left(\frac{E^*}{\sigma_{0.033}}\right)\right) + 29.267$$

Equation 11 - Dao et al.'s Π_1, which is used to solve for $\sigma_{0.033}$

Using these values, UNiT utilizes SciPy's `brentq` solver to solve the initial yield stress σ_y, which it can use to calculate further ϵ_y **(Equation 12)**. `brentq` essentially determines a zero of some curve that changes signs over a given interval (Scipy Optimize Brentq, n.d.).

$$\sigma_{0.033} = \sigma_y\left(1 + \left(\frac{E}{\sigma_y}\right) * 0.033\right)^n$$

Equation 12 - The equation used to solve for σ_y

UNiT then uses $\sigma_{0.033}$ and E^* in `brentq` to solve for the strain hardening exponent n, which must be in the range of 0 to 1 and is calculated through Dao et al.'s Π_2 formula **(Equation 13)**.

$$\Pi_2 = (-1.40557n^3 + 0.77526n^2 + 0.15830n - 0.06831)\left[\ln\left(\frac{E^*}{\sigma_{0.033}}\right)\right]^3 + (17.93006n^3 - 9.22091n^2 - 2.37733n + 0.86295)\left[\ln\left(\frac{E^*}{\sigma_{0.033}}\right)\right]^2 + (-79.99715n^3 + 40.55620n^2 + 9.00157n - 2.54543)\left[\ln\left(\frac{E^*}{\sigma_{0.033}}\right)\right] + (122.65069n^3 - 63.88418n^2 - 9.58936n + 6.20045)$$

Equation 13 – Dao et al.'s Π_2, which is used to solve for n

Lastly, UNiT begins creating the material's stress-strain curve by calculating the strength coefficient R **(Equation 14)**.

$$\sigma = R\epsilon^n$$

Equation 14 - Calculation of R, the strength coefficient, and the calculation of σ where $\sigma \geq \sigma_y$

This formula is also used in the following system of equations to create and output the final stress-strain curve for the data, which is plotted using the Matplotlib library **(Figure 8)**. Specifically, σ values less than σ_y are calculated as in **Equation 15**, and σ values greater than σ_y are calculated as above in **Equation 14**. ϵ values are simply created and plotted between 0 and 0.3 with a step value of 0.0001.

$$\sigma = E\epsilon$$

Equation 15 - The calculation for σ where $\sigma \leq \sigma_y$

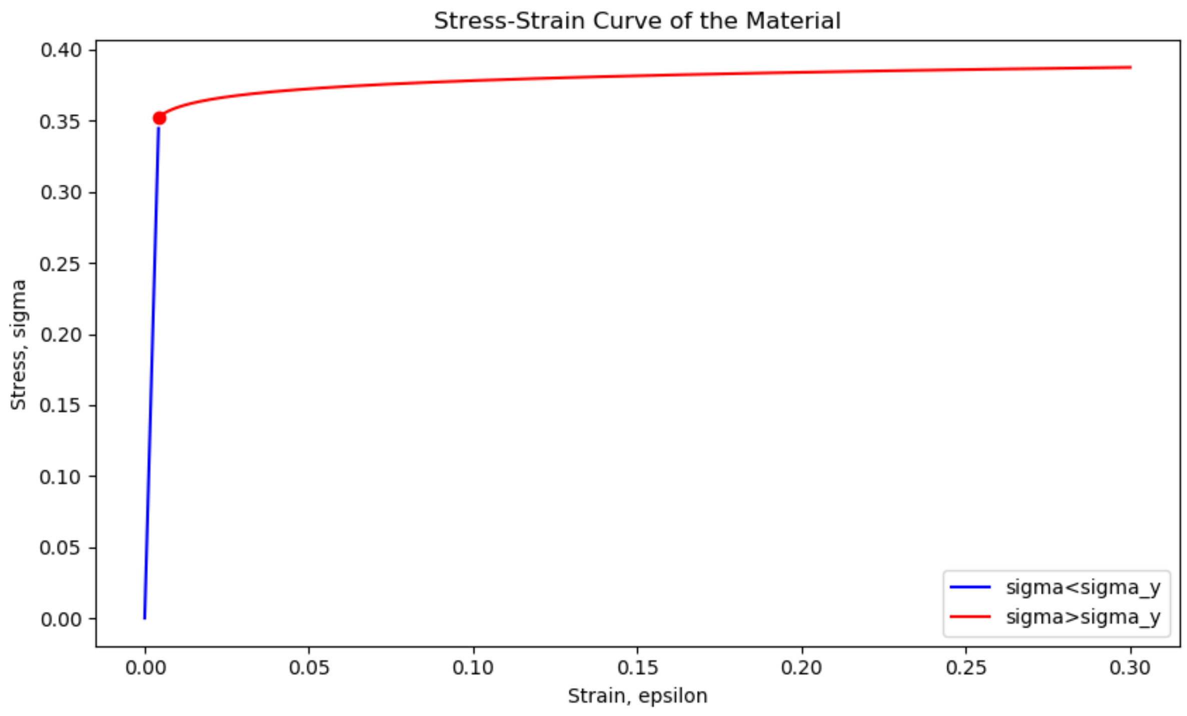


Figure 8 - The final stress-strain curve created by UNiT

Algorithm 3: Identifying the Relationship Between Work of Indentation and Hardness and Reduced Modulus

In their 2008 paper “Relation between the ratio of elastic work to the total work of indentation and the ratio of hardness to Young’s modulus for a perfect conical tip,” Chen and Bull identify several formulas to calculate the ratio of a material’s hardness, H, and its reduced modulus, E_R (Bull & Chen, 2009). As input, the algorithm requires load P, depth h, and stiffness S from nanoindentation data and the material’s Poisson’s ratio and the indenter’s half-angle. UNiT starts by calculating the loading and unloading curves by finding the maximum load and separating each half of the curve. UNiT also finds the unloading stiffness of the curve, S_U, which is simply the stiffness value at max load. The unloading curve is then translated in the negative direction to remove the “holding” period used during nanoindentation. From here, UNiT can use the load-depth curve to calculate several essential values. First, the work of indentation is calculated by integrating and adding the areas below the loading and unloading curves. **Figure 9** corresponds to the areas below the red and blue curves, respectively. Then, the value h_r is calculated by finding the X-intercept of the unloading curve. When calculating stress-strain values, this can be found by using a polynomial fit, such as that provided by Numpy’s Polynomial library (Numpy Polynomial Fit, n.d.). The above is summarized in **Figure 9**.

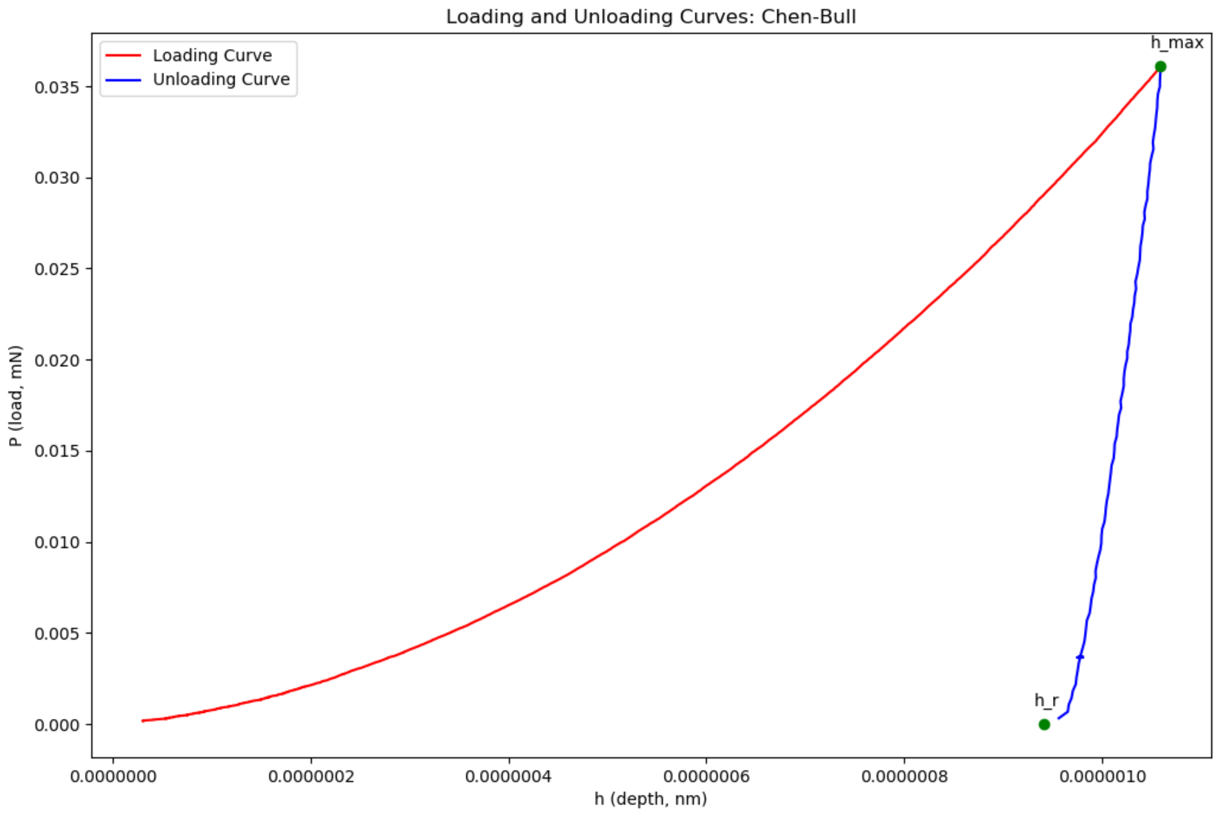


Figure 9 - Chen-Bull’s loading and unloading curves, with labels describing h_{max} and h_r

Next, UNiT uses `curve_fit` to solve for m, the exponent in the power-law described in **Equation 16**. P_m is the max load value, B is a parameter used in the power law, and δ values correspond to depth values described above.

$$P_m = B(\delta_m - \delta_r)^m$$

Equation 16 - Calculation of P_m, as described in Equation 4 of (Bull & Chen, 2009)

Finally, UNiT can calculate values for H and E_r of the material. To do this, it utilizes the following equations described in Chen and Bull's paper:

$$E_r = \frac{\left(1 - \frac{\delta_r}{\delta_m}\right) S_u^2 \cot\theta}{2\beta P_m[m - \left(1 - \frac{\delta_r}{\delta_m}\right)\epsilon]}$$

Equation 17 - Calculation of E_r, as described in Equation 28a of (Bull & Chen, 2009)

$$H = \frac{\left(1 - \frac{\delta_r}{\delta_m}\right)^2 S_u^2 \cot^2\theta}{\pi\beta P_m \left[m - \left(1 - \frac{\delta_r}{\delta_m}\right)\epsilon\right]^2}$$

Equation 18 - Calculation of H, as described in Equation 28b of (Bull & Chen, 2009)

Chen and Bull also describe other methods by which to calculate E_r and H (**Equations 19 and 20**, respectively). However, the team found that these methods were not as accurate as the methods described above, likely due to differences in calculating work of indentation. So, UNiT simply offers the methods above to calculate E_r and H.

$$E_r = \frac{W_e/W_t}{\frac{1.5\pi m}{1+m} - \pi\epsilon W_e/2W_t} \cot\theta \frac{4\beta P_m}{\pi S_u^2}$$

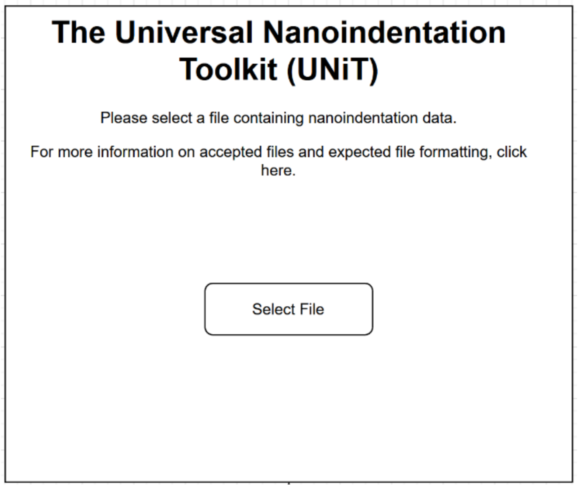
Equation 19 - Calculation of E_r, as described in Equation 29a of (Bull & Chen, 2009)

$$H = \left(\frac{\frac{W_e}{W_t}}{\frac{1.5\pi m}{1+m} - \frac{\pi\epsilon W_e}{2W_t}}\right)^2 \frac{4P_m}{\pi S_u^2}$$

Equation 20 - Calculation of H, as described in Equation 29b of (Bull & Chen, 2009)

User Interface Design

The user interface design process began by gathering requirements for the user interface (UI) and developing multiple mockups. When users first open UNiT, they are greeted by a popup requiring them to input an Excel sheet containing valid nanoindentation data (Figure 10). After selecting an Excel file, the user is prompted to select specific algorithms to run on their data and specify any necessary parameters for these algorithms (Figure 11). When an algorithm is selected in the dropdown at the top of the screen, a box appears with the relevant parameters that must be configured prior to running UNiT. Users can select multiple algorithms, and the page will update dynamically to account for each newly selected algorithm. After selecting algorithms and filling in their parameters, the user can simply click "Process" to perform the calculations on their data. Results are subsequently displayed on their screen, with each algorithm presenting both numerical and graphical output in its own section (Figure 12). Users are also given the option to save both the numerical and graphical results to their computer.



The Universal Nanoindentation Toolkit (UNiT)

Please select a file containing nanoindentation data.

For more information on accepted files and expected file formatting, click here.

Figure 10 - A mockup of UNiT's initial loading page

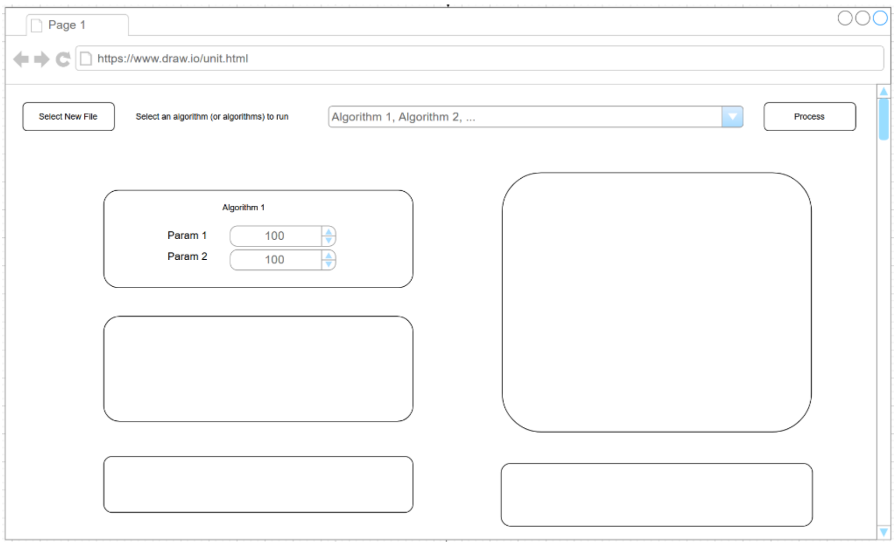

Page 1
https://www.draw.io/unit.html
Select New File
Select an algorithm (or algorithms) to run
Algorithm 1, Algorithm 2, ...
Process
Algorithm 1
Param 1
100
Param 2
100

Figure 11 - A mockup of UNiT's algorithm selection page

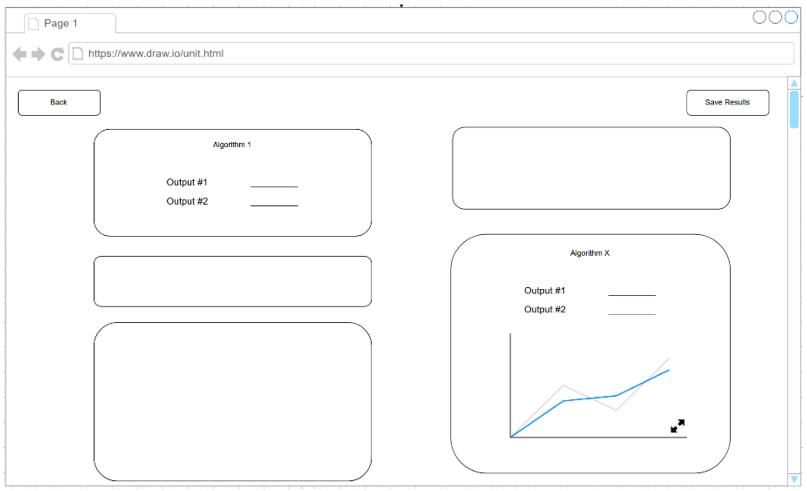


Figure 12 - A mockup of UNiT's Result Page

The user interface of UNiT was developed using TypeScript, React, and Material UI framework. These languages and frameworks were chosen due to their modern style, which would elevate UNiT over similar tools in nanoindentation research. Furthermore, each language and framework are also widely accessible and popular, allowing future users to edit and expand upon UNiT's interface as they see fit with relative ease.

Development and Deployment

The development process was broken into programming the user interface, algorithm-based backend, and API to connect the user interface and backend. Version control was handled through GitHub, and merge issues were avoided by separating user interface and backend work. At the time of publishing this paper, the repository containing UNiT was not made public as a result of ongoing publications beyond the scope of this project. After some consideration, a web browser was chosen to deploy the tool. The reasons for this were multifaceted. Those without a significant computing background may have difficulty installing or modifying a local version of the tool, and providing a web interface would make accessing the tool as simple as entering a URL in one's browser.

Verification

Implementations of algorithms were verified through various material properties tests. Specifically, team members skilled in using materials science equipment performed 3-5 indentation plastometry measurements per sample, 25-30 targeted indentation tests for each sample using nanoindentation equipment, hardness testing for each material, and computationally simulated tests. Data from relevant literature was also gathered to compare the results of each algorithm. A Plastometrex Indentation Plastometer was used to perform the plastometry measurement tests, for which it used a 1.0 mm radius ball tip and performed indentation on the material until it yielded, at which point results could be obtained. A KLA iMicro nanoindenter was used to perform nanoindentation tests and gather nanoindentation data.

The machine was equipped with a diamond Berkovich tip and flat punch tip and set to perform 25-30 targeted indents using the Oliver-Pharr contact method. All nanoindentation tests performed continuous stiffness (or dynamic) measurements. The data resulting from these tests were run through UNiT, where the results of each algorithm were compared to the results of each validation method. Each method of verification testing and relevant literature were used to support UNiT's results; for example, hardness measurements calculated by UNiT were backed up by manual hardness testing, indentation plastometry, computational simulation, and relevant literature. These methods have the added benefit of verifying the accuracy and proximity of the validation tests themselves.

RESULTS

Accuracy of Results

To evaluate the accuracy of the methods in UNiT, values collected from validation testing methods, which are outlined below, and literature are compared with results from the two algorithms in the program. A bar chart was chosen to visualize this comparison because it directly compares the several validation methods with UNiT's algorithms sorted by each material. For example, the main output from the Nix Gao method was the Vickers hardness of the material, and one of the main numerical outputs of Dao et al. was the material's yield strength. As such, Vickers hardness data was gathered from two different microhardness tests at 1 kgf and 0.1 kgf, flat punch nanoindentation, and literature. In addition, yield strength data from indentation plastometry, flat punch nanoindentation, and literature were used to compare with the Dao et al. method. Again, these values are considered the most well-known of the outputs and are therefore more significant when evaluated against other methods.

Figure 13 displays the first attempt at evaluating the Vickers hardness results output by UNiT's implementation of the Nix Gao method. Each value was within reason of the literature and tested values and in the correct order of magnitude between each material, which was the goal for UNiT. While this goal was reached, it became clear that the Nix Gao method initially overestimated the Vickers hardness of all the materials. Adjustments were then made to UNiT's algorithm implementation to counter the overestimation and provide more accurate results. Specifically, this was done by adjusting input depth and hardness values so that the Nix Gao method only utilized data where hardness is generally decreasing after a depth of 200 nm, as described in (Wright et al., 2013). In doing this, the estimations of Vickers hardness from Nix Gao were made notably more accurate, with only a slight overestimation **(Figure 14)**.

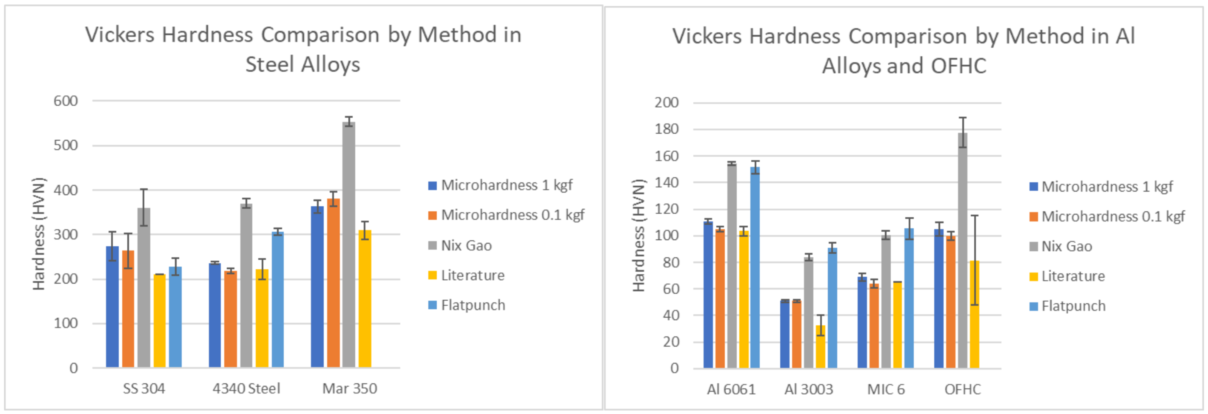


Figure 13 - Comparison of Vickers hardness across different testing methods and Nix Gao

It is also essential to look at the variance in values between the different validation methods. This is something to keep in mind when using test data, as it shows the value of using multiple testing methods for validation purposes. While each testing method is popular in materials science research and well-validated for finding Vickers hardness, they do not precisely align with each other. As expected, both Vickers microhardness values are nearly the same, with 0.1 kgf being slightly lower than 1 kgf. However, the flat punch data is higher than the microhardness tests, which is closer to the overestimated Nix Gao values. Again, this variance is expected to some degree between methods; however, it is interesting to see and note how they compare from a materials science standpoint.

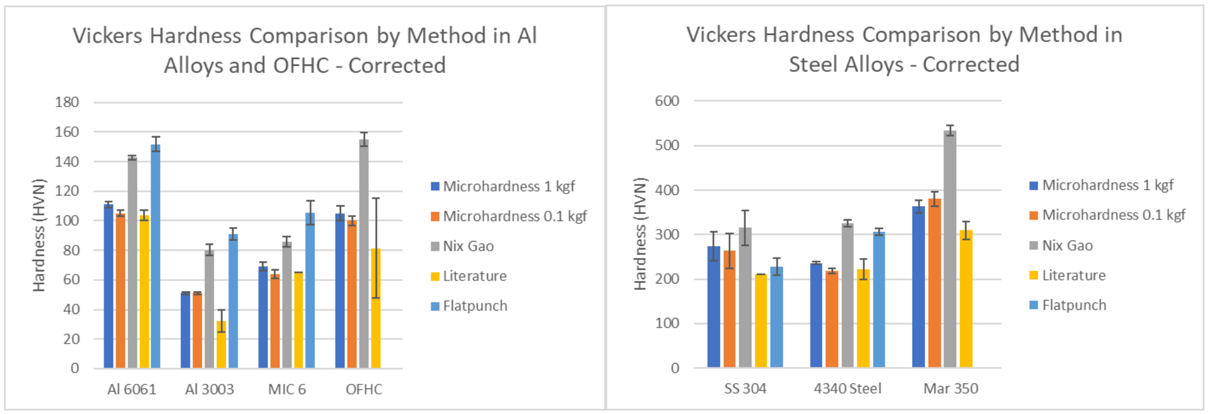


Figure 14 - Comparison of Vickers hardness across different testing methods and the corrected Nix Gao

UNiT's implementation of Dao et al.'s stress-strain calculation is evaluated below in the form of yield strength **(Figure 15)**. This method gave differing accuracies for each material and was not as consistent as the Vickers hardness results. For Aluminum 6061, 304 Stainless Steel, and Maraging 350 Steel, UNiT's Dao et al. provided fairly accurate results that matched closely to the validation values. However, for the remainder of the material set (i.e., Aluminum 3003, MIC 6, and 4340 Steel), the yield strength showed to be extremely low. For example, Al 3003 and MIC 6 had yield strengths less than 1 MPa according to Dao et al., while they both should have been in the 100-150 MPa range according to literature and other values. This shows a limitation of UNiT that is important to note for future work. So, while the Dao et al. method has relatively accurate findings in the areas it works well in, it falls short for specific materials. This stark inconsistency is easy to spot if a user checks the results against values from other methods, but it could lead to some issues if it goes through unnoticed.

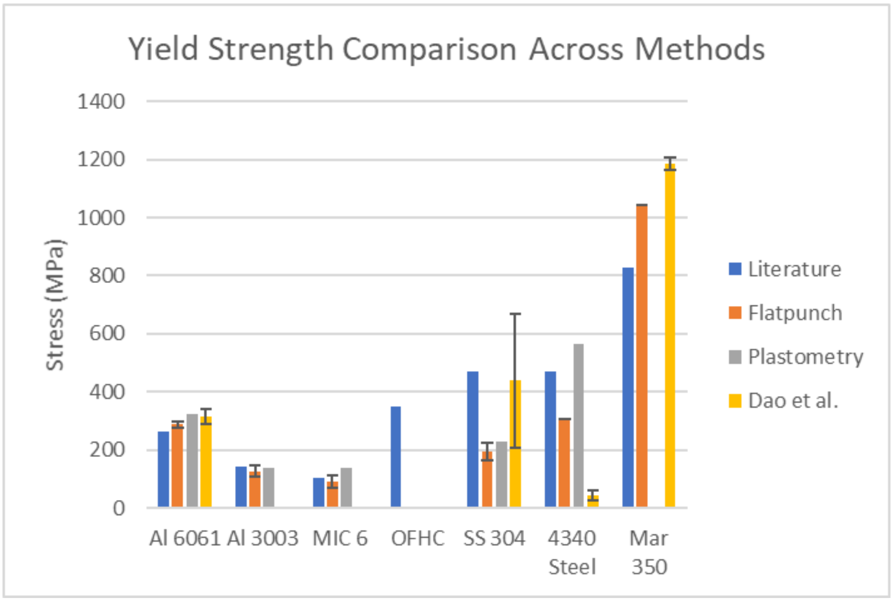


Figure 15 - Comparison of yield strength across different testing methods and UNiT's implementation of Dao et al

The Dao et al. method seemed to align best with the literature values. However, when only looking at the difference in validation values, the variance is apparent in 304 Stainless Steel, 4340 Steel, and Maraging 350 Steel. In many cases, the Dao et al. and literature data are more similar and larger than the other methods, while the flat punch data appears to be lower. Again, this shows the value in the multiple validation methods, as it gives insight into the variance of the tested values for material properties.

Reproducibility

Method	**Standard deviation**
Vickers 1	8.571
Vickers 0.1	10.143
Literature	12.429
FlatPunch	6.264
Nix Gao	11.560

Table 1 - Comparison of the standard deviation of Vickers hardness across different testing methods and UNiT's Nix Gao implementation

Method	**Standard deviation**
Flatpunch	11.707
Dao et al.	42.527

Dao et al. (SS304 outlier removed)	10.986

Table 2 - Comparison of the standard deviation of yield strength between flat punch and UNiT's Dao et al. implementation

The reproducibility of the results obtained through UNiT's Nix Gao implementation is on par with Vickers tests and the literature. However, the variation between each test is higher than that of the flat punch method. Both use an indenter tip of the same diameter but use different indenter tips. UNiT's implementation of Dao et al. had much higher variability precisely due to testing on a sample of 304 stainless steel, which appears to be an outlier. If this outlier is controlled, the variation falls very close to the flat punch method. To counter this variation, multiple tests were run on each material, and the average was used to compare to other methods and the literature.

User Testing

User testing was performed to determine the level of the Universal Nanoindentation Toolkit's (UNiT's) overall ease of use. UNiT team members administered user tests on four materials science researchers familiar with nanoindentation. These users were identified as ideal candidates due to their knowledge of nanoindentation and proximity to materials science research, meaning they would likely use UNiT if it were commercial software. Also, these users would benefit from having such a program because they do not have the solid technical computing backgrounds required to implement the algorithms that UNiT offers. User tests were administered remotely and involved the user manipulating the software through Zoom's remote-control feature while UNiT was open on a team member's device. Users were given a description of the software and its intention and were subsequently asked to perform various tasks using UNiT. While performing these tasks, users were given minimal instruction other than being told to perform said task. With this setup, users would be able to give feedback on the program without any influence from team members' instructions or opinions. In addition, other UNiT team members observed and recorded feedback from the users' actions and comments.

Participants were given the initial UNiT interface to start the user test and asked to select a specific Excel spreadsheet containing nanoindentation data. Then, users were asked to select an algorithm, input the parameters given to them, and initialize the test data processing. After this processing was complete, users were asked to save and view the algorithm's output on the Results page. Users were then asked to go back to the initial UNiT landing page to select an additional algorithm to process the data with. Users processed the data like before, this time running both algorithms simultaneously, then saved and viewed the output of each algorithm. Lastly, users were allowed to use the toolkit for any further testing they desired and the chance to give any general feedback they felt was necessary.

As a result of user testing, the UNiT team was able to identify areas for improvement in the software prototype. For example, several users pointed out that buttons used in the interface were slightly difficult to notice at first, and some users visibly struggled for a moment to locate the button required to process the data. As a result, UNiT would benefit from having larger, centered buttons that users can more easily interact with. Users also requested that various other information be displayed in more detail, such as adding more information about each algorithm and improving the data layout on the Results page. Lastly, regarding the user interface design, users requested that the display be more fine-tuned, such as having a less minimalistic design and using appropriate Greek lettering rather than the phonetic names for Greek letters. These results are summarized in **(Figure 16)**. Many of these areas were addressed by the UNiT team and added to the final version of the program.

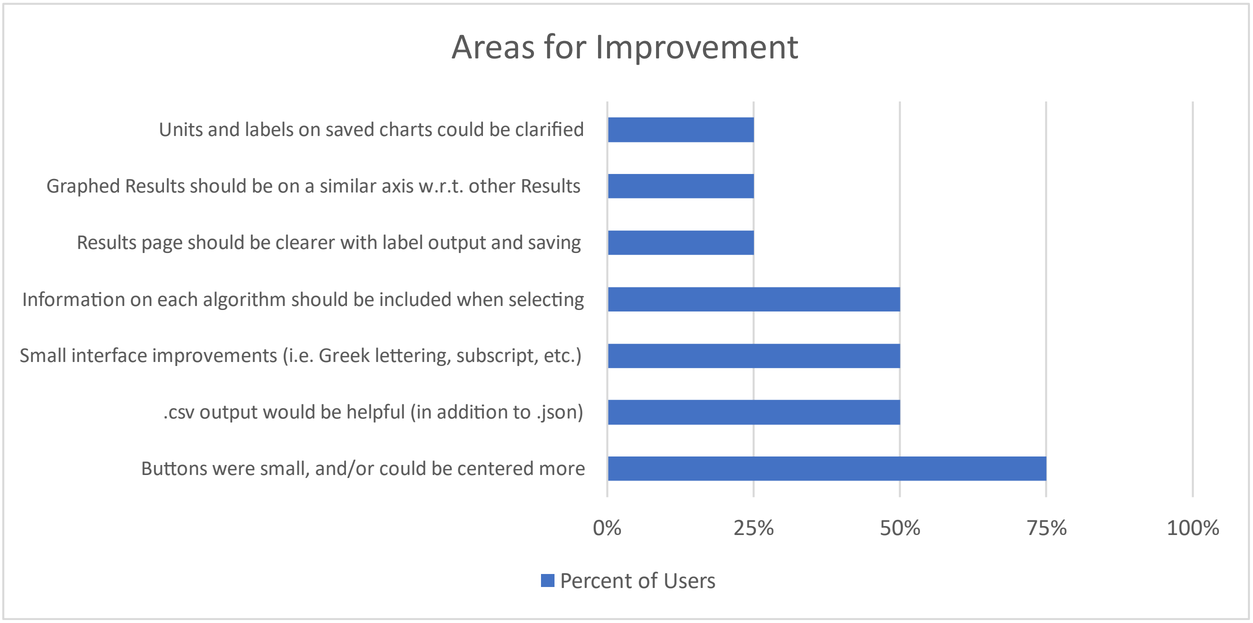


Figure 16 - Areas for improvement identified with user testing

User testing also identified multiple areas of success for UNiT. First and foremost, all users appreciated the feature allowing them to select and run multiple algorithms simultaneously. Alongside this fact, most users appreciated the program’s simple design and felt interacting with it was straightforward. As for calculating results from each algorithm, multiple users made the point that they liked the output format of the Results page and liked that they could save their results in raw data formats that were easy to analyze. Finally, users also recognized that using UNiT would save time compared to other methods (i.e., manual calculation) of obtaining results from the same algorithms. These results are summarized in **(Figure 17)**. The success areas match UNiT’s design philosophy of providing a straightforward interface that allows users to quickly calculate the results of various complex algorithms on nanoindentation data. User testing helped identify areas where UNiT could improve and areas where its design proved successful.

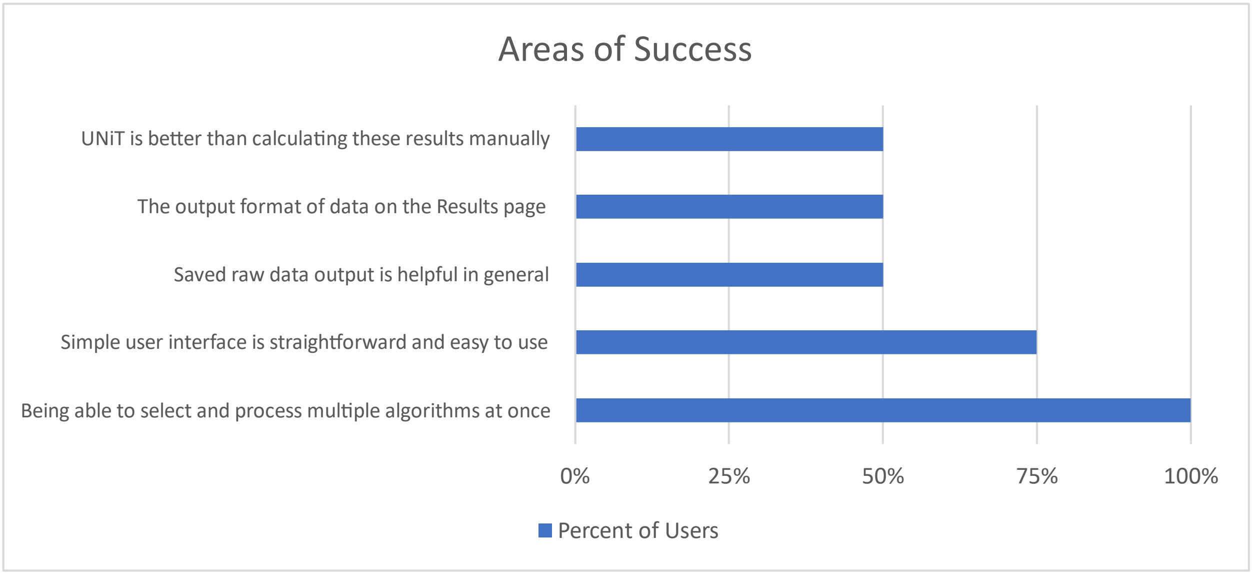


Figure 17 - Areas of success identified with user testing

DISCUSSION

The Universal Nanoindentation Toolkit (UNiT) achieved its goal of creating accurate implementations of complex algorithms that are generally unavailable to materials science researchers. User tests determined that while some changes to the interface would improve the overall user experience, users generally approved of several of the design decisions UNiT incorporated into its user interface. A simple, minimalist design meant that users could quickly understand the tool and process large amounts of nanoindentation data in ways that could not have feasibly been done before. The ability to save results in standard formats also proved helpful, as users could easily report on such data or otherwise utilize it how they please. Lastly, processing multiple algorithms simultaneously saved users a significant amount of time and improved accessibility to complex algorithms without a significant technical background. Overall, user tests demonstrated that UNiT's goal of providing a smooth user experience for materials science researchers was successful.

While the team used the average of the computed values when determining the accuracy of the algorithms included in UNiT, it is also worth considering the relevance of the standard deviation of these results. Materials are not entirely homogeneous in terms of their properties. Instead, they have an underlying microstructure that can vary across the material at a nanometer scale. Many traditional methods used to compute material properties operate on a much larger region of the sample material than nanoindentation. For example, nanoindenter tips are on the order of 10 μm while traditional methods for indentation testing operate on the 20 μm – 1 mm scale. Therefore, the standard deviation of the results can be used as an accurate measure of the distribution of material properties where the deviation is intrinsic to where the indentations were made. However, consideration of the underlying crystalline microstructure of materials must be taken when choosing the number of indentations. An adequate number of indentations can be selected on a per-material basis by increasing the number of indentations until statistical significance ($p<0.05$) in the resultant material properties is reached. Due to the deterministic nature of the underlying algorithms implemented in UNiT, the intramaterial variations in its output can be attributed to the materials' underlying microstructure and not the result of any variation in the algorithms themselves.

Overall, there was a lot to learn in building UNiT, and limitations were undoubtedly reached. Unfortunately, some goals that were initially set for this project could not be completed for several reasons, including unexpected issues with algorithms and time constraints. The inherent inaccuracies of the Chen Bull method, for example, limited it from being a contender for UNiT because it did not satisfy the level of accuracy that was needed. So, while it was implemented purely for example purposes, only the Nix Gao and Dao et al. methods were officially implemented into UNiT. It is hoped that more algorithms and improvements to UNiT can be made soon to counter the limitations faced in this project.

CONCLUSION

The main goals that were set out for the Universal Nanoindentation Toolkit (UNiT) were met through this project. Three algorithms were implemented into a user-friendly interface that allows a range of metals to be evaluated with minimal time and effort compared to traditional methods. These algorithms were validated through multiple traditional testing methods to ensure that the quality of UNiT's results was on par with scholarly sources. The culmination of these efforts is a fully functional software program that supplies materials scientists with a faster and easier way to calculate material properties for a range of materials. Future work can be done to improve the toolkit, which is why UNiT was also designed to be fully modularized, allowing for straightforward changes and additions. Overall, UNiT successfully meets its goals of providing materials scientists with a tool that allows them to efficiently analyze nanoindentation data through complex and previously inaccessible methods.

Future Work

While UNiT made significant steps towards providing its users with access to powerful algorithms, work remains to be done to make the toolkit ready for general use. First and foremost, users are currently required to install Python, Node.js, and several other libraries related to these languages. In future work, developers would likely want to find a way to deploy UNiT so users could easily access the tool through a browser or through downloading an application. Concerning improving UNiT's capabilities, developers would be able to add whatever algorithm they please, as UNiT provides modularity that allows for easy implementation of new algorithms. To further improve the processing power of these algorithms, developers would implement methods to allow the user to run algorithms on multiple test sheets, as currently UNiT only handles the first test sheet. When reading data, UNiT would also benefit from an in-house data cleaning suite to remove erroneous values from nanoindentation tests, such as values that are not in line with expected data for a test. Lastly, developers would improve the error handling capabilities of UNiT, as currently, the toolkit will display a non-descript error message if an error occurs while processing. For example, this could mean making algorithms still output valuable data even with a relatively minor error while processing. With all of this in mind, UNiT would become a considerably more robust and more resilient tool of greater use to the materials science research community.

CITATIONS

Bull, S. J., & Chen, J. (2009). Relation between the ratio of elastic work to the total work of indentationand the ratio of hardness to Young’s modulus for a perfect conical tip. Journal of Materials Research, 24, 590–598. https://doi.org/10.1557/jmr.2009.0086

Campbell, A. C., Grolich, P., & Slesinger, R. (2019). Niget: Nanoindentation general evaluation tool. SoftwareX. https://www.researchgate.net/publication/331705467_Niget_Nanoindentation_general_evaluation_tool

Dao, M., Chollacoop, N., Van Vliet, K. J., Venkatesh, T. A., & Suresh, S. (2001). Computational modeling of the forward and reverse problems in instrumented sharp indentation. Acta Materialia, 49(19), 3899–3918. https://doi.org/10.1016/S1359-6454(01)00295-6

Haušild, P. (2021). On the breakdown of the Nix-Gao model for indentation size effect. Philosophical Magazine, 101(4), 420–434. https://doi.org/10.1080/14786435.2020.1841916

Hill, J., Mulholland, G., Persson, K., Seshadri, R., Wolverton, C., & Meredig, B. (2016). Materials science with large-scale data and informatics: Unlocking new opportunities. MRS Bulletin, 41(5), 399-409. http://dx.doi.org.ezpv7-web-p-u01.wpi.edu/10.1557/mrs.2016.93

Matplotlib—Visualization with python. (n.d.). Retrieved December 16, 2021, from https://matplotlib.org/

Mercier, D. (2018). PopIn Toolbox documentation. https://doi.org/10.13140/RG.2.2.33497.16481

Nanomechanics Inc. (2018). NanoBlitzTM 3D Mechanical Property Mapping NanoBlitzTM 4D Mechanical Property Tomography Operating Instructions : Vol. R2.0. Nanomechanics, Inc.

Nix, W. D., & Gao, H. (1998). Indentation size effects in crystalline materials: A law for strain gradient plasticity. Journal of the Mechanics and Physics of Solids, 46(3), 411–425. https://doi.org/10.1016/S0022-5096(97)00086-0

Pandas documentation—Pandas 1. 3. 5 documentation. (n.d.). Retrieved December 16, 2021, from https://pandas.pydata.org/pandas-docs/stable/index.html

Plante, R. L., Becker, C. A., Medina-Smith, A., Brady, K., Dima, A., Long, B., … Hanisch, R. J. (2021). Implementing a Registry Federation for Materials Science Data Discovery. Data Science Journal, 20(1), 15. DOI: http://doi.org/10.5334/dsj-2021-015

Scikit-learn: Machine learning in python—Scikit-learn 1. 0. 1 documentation. (n.d.). Retrieved December 15, 2021, from https://scikit-learn.org/stable/index.html

SciPy. (n.d.). Retrieved December 15, 2021, from https://scipy.org/

Scipy. Optimize. Brentq—Scipy v1. 8. 0 manual. (n.d.). Retrieved February 27, 2022, from https://docs.scipy.org/doc/scipy/reference/generated/scipy.optimize.brentq.html

SciPy. Optimize. Curve_fit—Scipy v1. 7. 1 manual. (n.d.). Retrieved December 16, 2021, from https://docs.scipy.org/doc/scipy/reference/generated/scipy.optimize.curve_fit.html

Scipy. Optimize. Fsolve—Scipy v1. 8. 0 manual. (n.d.). Retrieved February 27, 2022, from https://docs.scipy.org/doc/scipy/reference/generated/scipy.optimize.fsolve.html

Wright, W. J., Feng, G., & Nix, W. D. (2013). A Laboratory Experiment Using Nanoindentation to Demonstrate the Indentation Size Effect. Journal of Materials Education, 35(5–6), 135–144.