BPR MCS
Bulletin of Pioneering Researches of Medical and Clinical Science

Bulletin of Pioneering Researches of Medical and Clinical Science

Available online: https://bprmcs.com
2025 | Volume 5 | Issue 1 | Page: 228-241

Improving High-Strain-Rate Measurements via Stress Wave Dispersion Compensation in SHPB Experiments

Ahmed El-Kholy[1]*, Nour Abdelrahman[1], Karim Hassan[2]

[1]Department of Clinical Medicine and Biomedical Sciences, Alexandria University, Alexandria, Egypt.
[2]Department of Medical Research Systems, Ain Shams University, Cairo, Egypt.

Abstract

The phenomenon of stress wave dispersion can attenuate or corrupt crucial high-frequency components during material testing at high strain rates or blast-loading trials. This study aims to demonstrate the value of applying dispersion compensation in split-Hopkinson pressure bar experiments conducted under various test conditions. For this purpose, a new computational framework, designated `SHPB_Processing.py`, is constructed. After guiding the reader through an operational overview of `SHPB_Processing.py`'s functionalities, the tool is deployed to analyze experimental records from split-Hopkinson pressure bar examinations of aluminum, sand, and kaolin clay specimens across a range of test configurations. A side-by-side evaluation of dispersion-corrected data and those obtained through straightforward time-alignment from SHPB trials reveals that taking dispersion into account suppresses artificial oscillations and refines the derived measurement at the specimen's leading edge. The precision of the stress and strain outputs produced by its use highlights its critical role, as evidenced by the stark discrepancy between its deployment and its absence. This carries considerable weight for the credibility, exactitude, and caliber of the outcomes. Hence, moving forward, this instrument can be employed in any cylindrical-bar strain-rate testing context that demands dispersion rectification, confinement, or stress equilibrium appraisal.

Keywords: Signal processing, Dispersion correction, High-strain-rate testing, Stress waves, Split-Hopkinson pressure bar, Material applications

Corresponding author: Ahmed El-Kholy
E-mail: ahmed.elkholy@outlook.com

Received: 15 January 2025
Revised: 30 March 2025
Accepted: 05 April 2025

How to Cite This Article: El-Kholy A, Abdelrahman N, Hassan K. Improving High-Strain-Rate Measurements via Stress Wave Dispersion Compensation in SHPB Experiments. Bull Pioneer Res Med Clin Sci. 2025;5(1):228-41. https://doi.org/10.51847/lZyFyppwlv

Introduction

Historically, a Hopkinson pressure bar (HPB) serves to gauge the transient pulse originating from the strike of proximate blast occurrences or projectiles. The split-Hopkinson pressure bar (SHPB), commonly called the Kolsky bar, has been widely adopted for characterizing the dynamic behavior of materials—producing curves such as stress–strain and strain rate–strain—across a diverse set of substances at strain rates between 10^2 and 10^4 s^{-1}. The elastic wave's profile in both SHPB and HPB evolves as it advances; this behavior is termed dispersion [1].

Seen through the lens of medium particle kinematics, the physical root of dispersion lies in the inertial effects of lateral movement coupled to the axial perturbation. From the vantage point of wave motion, a high-frequency constituent of the complete elastic wave propagates at a lower speed than its low-frequency counterpart [2].

Typically, the wave signature is captured at an intermediate axial station along the bar via strain gauges. In HPB configurations, the region of concern is the bar's

This work is licensed under the Creative Commons Attribution 4.0 International License.

foremost face, where the impact pulse enters; in SHPB setups, the focus is the specimen's position. Accordingly, the registered wave profiles in HPB and SHPB must be adjusted to reconstruct the wave profiles at the relevant positions, a process referred to as dispersion correction [1, 3].

The one-dimensional wave model presumes that every longitudinal wave within the bar advances at an unchanging speed, $c0$. It further supposes that transverse slices of the bar remain planar and that stress across any such slice is evenly spread. In practice, however, as the wave progresses, axial strains give rise to radial bulging and shrinking, governed by Poisson's ratio of the bar material. This radial displacement upsets the uniformity of stress throughout the bar's cross-section, causing initially plane sections to distort [2].

The results of this departure from idealized assumptions are expressed in the three-dimensional wave formulations proposed by Pochhammer [4] and Chree [5], which were subsequently applied to longitudinal waves in cylindrical bars by Bancroft [6]. In place of a uniform progression at speed $c0$, longitudinal waves were proven to advance at a particular velocity $c\omega$, which is a function of the wavelength, the bar's diameter, the one-dimensional wave speed, and Poisson's ratio, as captured in Eq. 1:

$$(x-1)^2\varphi(ha)-(\beta x-1)\left[x-\varphi(\kappa a)\right]=0 \qquad (1)$$

Where

$$\begin{aligned}\beta &= (1-2\nu)/(1-\nu)\\ x &= (c_\omega/c_0)^2(1+\nu)\\ h &= \gamma(\beta x-1)^{\frac{1}{2}}\\ \kappa &= \gamma(2x-1)^{\frac{1}{2}}\\ \varphi(y) &= yJ_0(y)/J_1(y)\end{aligned}$$

And where $c\omega$ denotes the phase velocity, $c0$ denotes the one-dimensional elastic wave velocity, a is the bar radius, ν is Poisson's ratio, γ is the wave number, $2\pi/\lambda$, λ is the wavelength, and $Jn(\ )$ is the Bessel function of the first kind, of the order n.

This relation yields an infinite number of roots, each associated with a distinct mode of propagation within the bar, as shown in **Figure 1**. The implication is that lower-frequency waves advance at a speed roughly matching $c0$. In contrast, the phase velocity diminishes as frequency climbs, most notably once the wavelength becomes comparable to the bar's diameter.

The intricate wave signals generated during an SHPB experiment span a wide range of frequencies. This frequency dependency causes stress to spread out as it traverses the bar. The effect is visualized in **Figure 2**, which captures the dispersion of a trapezoidal pulse in a stainless-steel pressure bar. This pulse dispersion coincides with frequency-dependent alterations in both stress and strain over the bar's cross-section [7]. As **Figure 3** illustrates, an increase in the forcing frequency leads to a diminution of the strains sensed at the bar's periphery in contrast to those detected at its central axis. These consequences for phase velocity and magnitude entail that a strain history logged at the bar's exterior may fall short of faithfully portraying the averaged strain and stress at the bar terminus that interfaces with the specimen, positioned at a distance.

Conventional methodology, as described by Gray [8] in the ASM handbook, treats a straightforward temporal shift of all SHPB-collected signals as an adequate approach; however, this practice can lead to substantial flaws and imprecision.

Earlier investigations by Shin [1, 3, 9] yielded MATLAB and Excel scripts centered on dispersion for post-processing SHPB measurements. Those routines targeted phase-velocity rectification rather than amplitude adjustments. Though suitable for a range of scenarios, tests rich in high-frequency content or involving large bar diameters will exhibit significant stress and strain non-uniformities across the cross-section, rendering amplitude correction valuable for a faithful assessment of the specimen response [10].

The present undertaking aims to engineer an algorithm capable of addressing the challenges associated with dispersion in SHPB testing. To that end, the foundational theory underpinning dispersion correction, stress wave equilibrium, and confinement evaluation in SHPB trials is first examined. Subsequently, the aforementioned resource, `SHPB_Processing.py`, is laid out along with its entire suite of capabilities and subroutines. To close, it is applied to the collected SHPB test records, confirming its tangible utility.

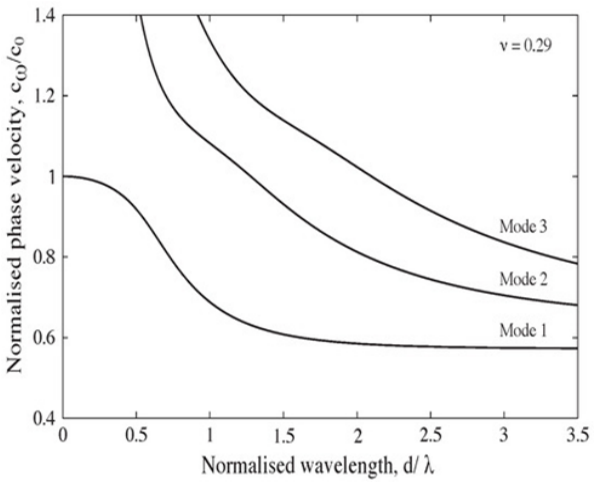


Figure 1. Dependence of phase velocity on wavelength for the first three longitudinal wave propagation modes within a cylindrical stainless-steel bar, assuming ν = 0.29 [7].

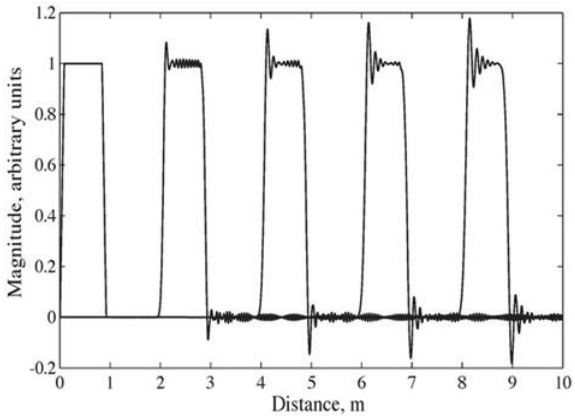


Figure 2. Broadening of a trapezoidal waveform as it travels through a cylindrical stainless-steel pressure bar, with readings taken every 2 m [7].

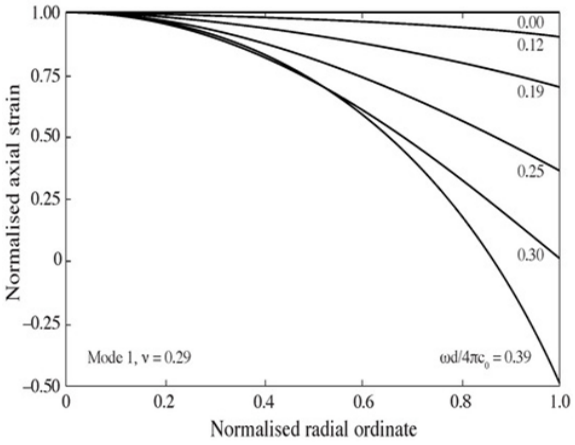


Figure 3. Profile of axial strain across the cross section of a stainless-steel ($\nu = 0.29$) bar for a single-frequency forcing function of infinite duration [7], adapted from Tyas and Watson [11].

Compensating for dispersion in SHPB testing

Once frequencies become elevated ($a/\lambda > 0.05$), the previously described inaccuracies become substantial; however, they can be corrected using the approach detailed by Tyas and Pope [10], which involves modifying the magnitude and phase angle of each frequency component in the signal.

Adjusting the phase angle

The initial operation performed on SHPB signals is the realignment of phase angles to counteract the spreading each frequency component experiences over the span between the strain gauge and the bar's extremity. Following Gorham [12] and Follansbee and Frantz [13], this is carried out by first calculating the phase velocity $c\omega$ for every component through Bancroft [6] relation (Eq. 1), and then imposing a phase shift, $\theta'\omega$, as expressed in Eq. 2:

$$\theta'_{\omega} = \left(\frac{c_0}{c_{\omega}} - 1\right)\frac{\omega z}{c_0} \tag{2}$$

Where ω stands for the angular frequency of the component, and z denotes the distance over which the compensation is applied, with the positive sense aligned with the direction the wave travels.

Barr *et al*. [14] conducted studies to evaluate how energy is distributed among higher-order propagation modes, ultimately finding that the frequency composition encountered in conventional SHPB work requires attention only to the first propagation mode.

Adjusting the amplitude

The second operation on SHPB signals consists of applying scaling multipliers to the frequency-component amplitudes. Tyas and Watson [11] introduced the multipliers $M1$ and $M2$ to account for radial non-uniformity in strain and Young's modulus, respectively, building on Davies [15] exploration of such radial effects. With these multipliers, a strain reading collected on the outer surface of the bar can serve to infer the mean axial stress and strain prevailing over the entire cross section. During an SHPB trial, the phase angle (dispersion) correction transforms a surface-level measurement obtained at the gauge position into a surface-level measurement at the plane where the specimen meets the bar; the amplitude correction then converts this surface-level quantity into the average strain and stress acting across the whole specimen interface.

The multipliers are defined in Eq. 3 and 4 given below:

$$M_1 = \frac{2\left(1 + \frac{1-\beta x}{x-1}\right)}{\varphi(ha) + \frac{1-\beta x}{x-1}\varphi(\kappa a)} \tag{3}$$

$$M_2 = E\left(\frac{c_{\omega}}{c_0}\right)^2 \tag{4}$$

Where the meanings attached to the symbols in Eq. 3 and 4 remain identical to those in Eq. 1, with E denoting Young's modulus.

Figure 4 charts the evolution of $M1$ and $M2$ with normalized wavelength for a stainless-steel bar whose Poisson's ratio equals 0.29. Because $M1$ exhibits a discontinuity at $a/\lambda = 0.375$—the very location where the surface-measured strain becomes zero—the inverse of $M1$ is plotted instead; at frequencies higher still, the sensed strain flips polarity compared to the mean cross-sectional behavior. Given that the corrective measures required at $a/\lambda = 0.375$ involve multiplying a tiny signal by a correction factor of extreme magnitude, any noise present in the signal stands a strong chance of severely undermining the reliability of the result. This places a practical boundary on the frequency interval open to

correction: according to Tyas and Watson [11], the method remains usable at normalized wavelengths staying beneath $a/\lambda \approx 0.3$.

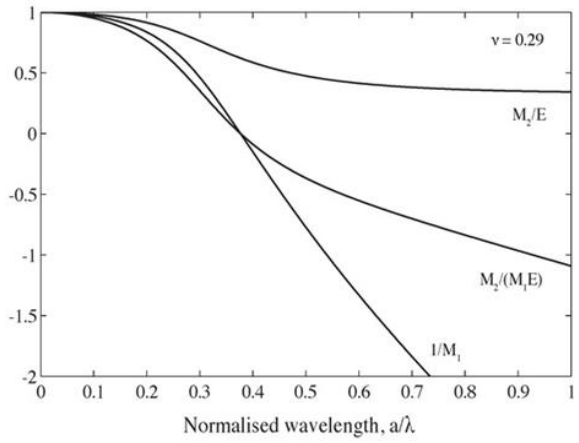


Figure 4. Change in the factors $M1$ and $M2$ for a cylindrical stainless-steel bar with $\nu = 0.29$ [7].

SHPB_Processing.py

`SHPB_Processing.py` constitutes a freely available Python tool aimed at the treatment of high-strain-rate signals originating from SHPB setups. Bundled within it is a subroutine named `dispersion.py`, tailored specifically to handle unprocessed strain gauge recordings by means of dispersion compensation (Section 4).

Designed to ingest strain gauge outputs collected during high-strain-rate SHPB trials, the function `SHPB_Processing.py`—once the user supplies the extra input quantities cataloged in **Table 1**—resolves the axial stress history, and, should confinement be stipulated, the radial stress history, together with the evolution of strain and strain rate across the impact event, plus further metrics derivable from those results.

Table 1. Input and output quantities utilized within `SHPB_Processing.py`.

Parameter	Explanation
csv_path	The directory path points to the CSV file that contains the raw experimental dataset.
sample_data	Array specifying specimen properties, including initial length, total mass, and dry mass, formatted as [initial length, mass, dry mass].
confinement	Type of confinement applied during the test, such as 'None', 'Ring', or 'Reservoir'.
signal_channels	Oscilloscope channel indices used for capturing raw signals, given as [in_bar_gauge_channel, out_bar_gauge_channel, ring_gauge_channel, or reservoir_gauge_channel].
signal_amp	Amplification factors applied to strain gauge signals, expressed as [in_bar_gauge_amp, out_bar_gauge_amp, ring_gauge_amp].
disp_correction	Option to apply either dispersion correction or a simple temporal shift to the signal data; "True" activates dispersion correction using dispersion.py.
alignment	Method used to align stress waves at specimen interfaces. 'start' aligns the onset of incident and transmitted pulses, 'end' aligns their endpoints, and 'mid' aligns their midpoint in time. Numeric values greater than 1 align pulse peaks to specified times, while values between 0 and 1 align based on a fraction of the peak amplitude.
speedtrap	Indicates whether speed trap data is used to compute striker bar velocity; set to 'True' to enable velocity calculation.
Processed data folder	Output directory containing all processed CSV files along with a test log for tracking processing history.

Launching this algorithm requires executing the command line statement below:

```
SHPB_Processing(csv_path,sample_data,confinement,signal_channels,
signal_amp,disp_correction,alignment,speedtrap)
```

The most effective workflow for putting this function to use is elaborated in the text that follows, with a compact visual guide supplied in **Figure 5**:

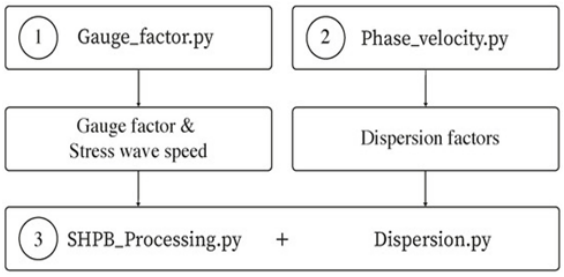


Figure 5. Flowchart summarising the steps for the efficient execution of `SHPB_Processing.py`.

- Establish the stress wave celerity and the gauge factors belonging to the cylindrical bars employed during SHPB work by running the `gauge_factor.py` script, hosted on GitHub and ORDA [16].
- Make use of `phase_velocity.py` to derive the dispersion coefficients required for the dispersion rectification of the captured SHPB waveforms via `dispersion.py`, drawing upon the material attributes of the cylindrical bar adopted for SHPB experimentation. The `phase_velocity.py` script is likewise accessible through GitHub and ORDA.
- At this stage, the main routine `SHPB_Processing.py` is prepped for launch, together with `dispersion.py`, to reduce the SHPB experimental data with dispersion correction activated competently, guided by the chosen input settings. All generated outputs are deposited into a pre-assigned processed data directory. `Dispersion.py` can be retrieved from GitHub and ORDA.

The entire source listing for `SHPB_Processing.py` resides on GitHub and ORDA [17].

The logic that governs the function can be condensed into the following sequence of operations:

- Raw oscilloscope traces produced by the SHPB strain gauges are loaded into memory.
- The striker bar's speed is extracted from the unprocessed speed trap measurements.
- Through pulse identification and reformatting of the signals, the raw data file is conditioned for the subsequent correction and confinement scrutiny.
- The selected correction strategy ('True' invoking dispersion correction, 'False' invoking a plain temporal translation) and confinement category ('None', 'Ring', or 'Reservoir') are imposed on the collected strain traces, as dictated by the user's input choices.
- Exploiting the trigger and the known wave propagation velocity of the bars utilized in the SHPB configuration, the incident, reflected, and transmitted pulses are singled out.
- The pulse is deemed to have ceased at the instant the specimen strain culminates at its peak.
- The dispersion-adjusted stresses and strains belonging to each wave component are evaluated through `dispersion.py`, whose inner workings are laid out further below. When a simple time shift is requested, rudimentary signal reshuffling is performed instead.
- Axial stresses and strains developing inside the specimen are quantified by combining the incident, reflected, and transmitted wave signatures.
- The deformation experienced by the specimen is inferred from the displacement of the pressure bars, using the strain gauge indications as the basis.
- Depending on the form of confinement nominated—'None', 'Ring', or 'Reservoir'—one of the following three paths is followed:
  a. If the confinement type is set to 'None' for an SHPB experiment, radial stresses and radial strains are not computed for the specimen.
  b. If the confinement type is set to 'Ring', the radial stress and radial strain imparted to the specimen are back-calculated from the circumferential strain registered in the confining ring, employing thick-walled pipe theory.
  c. If the confinement type is set to 'Reservoir', the pressure values logged by the transducer inside the reservoir serve as the basis for computing the radial stress and strain experienced by the specimen.
- For both the 'Ring' and 'Reservoir' confinement selections, the specimen's bulk density and its dry density are evaluated.
- Every result set is written out in CSV format into the Processed data folder, accompanied by the corresponding test log.

Dispersion.py

A python function for dispersion correction

To automate the process of adjusting both phase angles and amplitudes of SHPB waveforms within the overarching `SHPB_Processing.py` framework, an open-source Python utility named `Dispersion.py` was built. In place of elementary temporal shifting of entire signals, the code isolates and modifies each frequency band separately. An exposition of what the tool delivers follows below, and the unredacted source of `dispersion` is provided.py, togeth `py`, together with its subsidiary routine, can be obtained from GitHub and ORDA [18].

Frequency domain in python

Transforming a time history into its frequency-domain counterpart is accomplished through the fast Fourier transform (FFT). Conceptually, the method reconstructs any waveform as a sum of sinusoids, all differing in frequency and strength. In Python, this operation relies on the `fft` routine in the `numpy` library; when supplied with a uniformly spaced input, it returns a complex-valued array ordered by frequency, with each entry of the form z = zr + izi, encoding both magnitude and phase angle. At a chosen frequency, the associated amplitude A comes from Eq. 5 and the phase angle θ from Eq. 6, given here:

$$A = \sqrt{z_r^2 + z_i^2} \quad (5)$$

$$\theta = \tan^{-1}\left(\frac{z_i}{z_r}\right) \quad (6)$$

These ideas are given geometric interpretations in **Figure 6a**, which places z and its complex conjugate $\bar{z}$ in the complex plane, and in **Figure 6b**, which illustrates how the same pair of numbers defines the amplitude and phase of a single sinusoid.

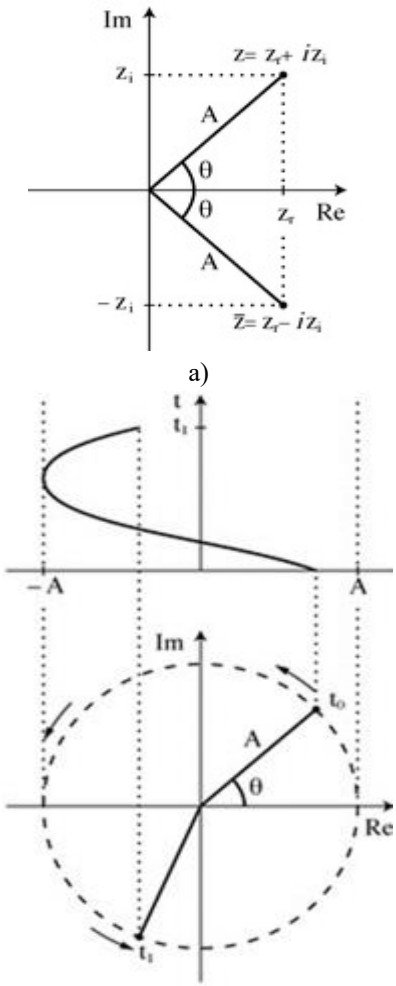


Figure 6. Depiction of a Fourier component z within the complex plane: (a) how it links to amplitude and phase angle, and (b) how it describes a sinusoid [7].

Once the amplitude and phase have been suitably modified, the Fourier component can be reassembled via the formula labeled Eq. 7, shown below:

$$z = A\cos(\theta) + iA\sin(\theta) = Ae^{i\theta} \quad (7)$$

Correction bandwidth

Computing the discrete Fourier transform (DFT) efficiently is precisely what the FFT achieves. The DFT yields frequency contributions at a countable set of frequencies, the spacing of which hinges on how fast the original record was sampled and how many points it contains. Should a trace be captured N times at a sampling frequency f, the resolvable frequency that sits at the bottom of the range equals f/N, representing a wave that exactly fits once inside the observation window **(Figure 7a)**. At the opposite extreme sits the Nyquist frequency, f/2, which marks the ceiling of what can be faithfully read **(Figure 7b)**. This upper bound stems from the sampling requirement that at least two data points per cycle be present to prevent aliasing, as shown in **Figure 7c**. When the sampling density drops too low, a pair of distinct sinusoids can both pass through the identical set of recorded points. For the tests discussed here, the oscilloscope's sampling rate sets f/2 at 500 kHz, thereby fixing the maximum recoverable frequency; the sharpness of the frequency grid, however, can be increased by enlarging N, whether by prolonging the recording itself or by appending zeroes to the captured signal.

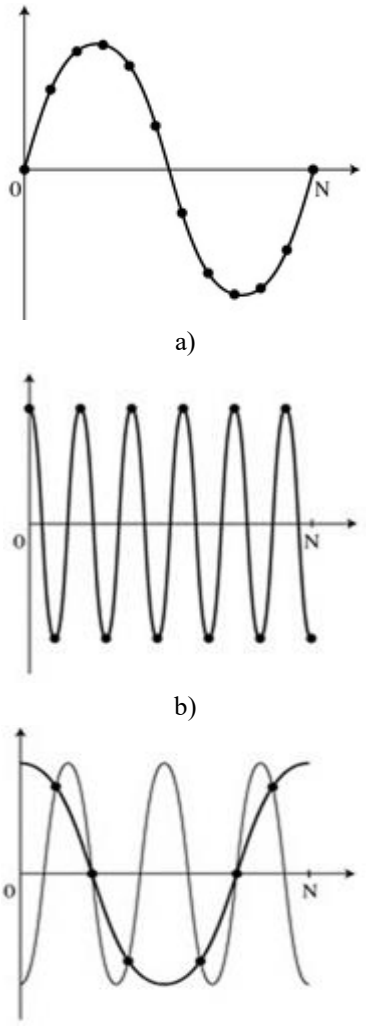


Figure 7. Constraints on frequency resolution in the FFT: (a) lowest resolvable frequency, where the sampling window accommodates exactly one wavelength, (b) highest resolvable frequency, where

each period is represented by only two samples, and (c) aliasing at frequencies beyond this, where multiple sinusoids can be drawn through the very same data [7].

When the `fft` routine receives an N-point time-domain series x(t), it hands back an N-point frequency-domain series X(ω). Because of the aliasing just discussed, the trailing half of X(ω) is a mirror image of the leading half—the complex conjugate reflected across the Nyquist frequency—as sketched in **Figure 8**. Practically, this means corrections can be restricted to the first N/2 + 1 bins alone, after which the adjusted bins are mirrored to repopulate the full vector.

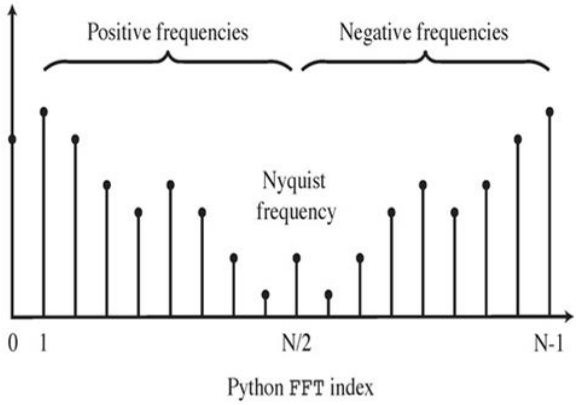


Figure 8. How the frequency-domain vector returned by Python's `fft` is arranged. The Nyquist frequency, at index N/2, defines the highest usable frequency. Beyond it, the second half of the vector repeats the complex conjugates of the first half [7].

An additional frequency limitation arises from a phenomenon already mentioned in Section 2.2: at wavelengths shorter than a/$\lambda \approx 0.3$, the strains picked up on the bar's circumference become vanishingly small. To give a concrete illustration, in the SHPB system used here, fitted with a 25 mm-diameter stainless-steel bar, the usable correction window spans only from 39 μHz to 94 kHz. **Figure 9** presents a frequency-domain portrait of a representative experimental incident pulse, shown as a modified periodogram. The ordinate is logarithmic; a 10 dB increase corresponds to a tenfold increase in signal power. Consistent with the preceding discussion, power at the bar's surface descends steeply toward zero across the band from 94 kHz to 110 kHz. Because dispersion correction is not viable beyond 94 kHz in this setup, a low-pass filter is applied first to remove everything above that threshold.

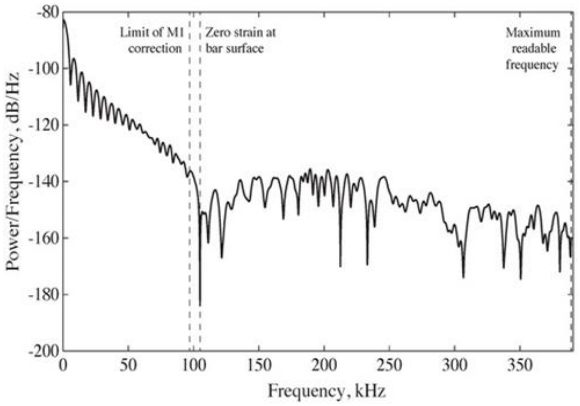


Figure 9. Modified periodogram showing power spectral density for a measured incident wave, sourced from a 25 mm stainless-steel bar with a Poisson's ratio of 0.29, overlaid with the upper frequency bounds imposed by the strain gauge data and the FFT [7].

As evident from **Figure 10**, the energy contained at those excised frequencies is diminished by orders of magnitude; accordingly, the filtration discards hardly any meaningful information.

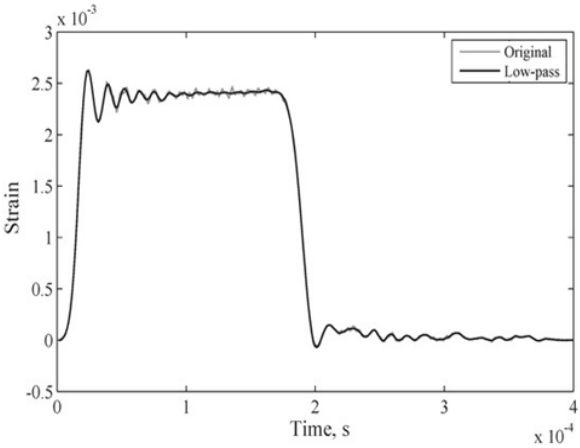


Figure 10. The incident waveform recorded in this experimental program is plotted alongside the low-pass-filtered trace, which removes content above 94 kHz [7].

Operation of dispersion.py

When the user enables the dispersion correction pathway inside `SHPB_Processing.py`, a subordinate open-source Python module—`dispersion.py`—is summoned to assist in reducing the SHPB waveforms collected during the experiments.

The module `dispersion.py` was constructed to eliminate manual intervention by automatically transferring the phase and amplitude adjustment coefficients produced by `dispersion_factors.py` onto the raw pressure bar traces. Its approach is to break each signal into its spectral components and then correct the accumulated effects of dispersion over the specified propagation span.

Running `dispersion.py` necessarily requires the companion script `dispersion_factors.py` to be present. Once the incident, reflected, and transmitted pulses have been individually identified and extracted, `dispersion.py` is tasked with reconstructing the stress and strain that were acting at the boundary between bar and specimen for each of those three waves. This is triggered by the invocation reproduced further down, which references the input and output parameters listed in **Table 2**. Poisson's ratio ν can generally be looked up from the manufacturer's material datasheets. At the same time, the one-dimensional wave speed $c0$ may be deduced by monitoring the oscillation of a predominantly low-frequency pulse inside a bar whose length is already established. Where direct measurements are inconvenient, iterative solution techniques, such as the one formulated by Shin [1], offer a practical alternative.

Table 2. Summary of arguments passed into and returned from `dispersion.py`.

Parameter	Description
x	Time-domain strain signal with zero-padding, represented as a 1×N numerical array.
fs	Sampling rate of the signal, expressed in Hz.
a	Radius of the bar, measured in meters (m).
c_0	One-dimensional wave propagation velocity within the bar, in m/s.
E	Young's modulus of the bar material, given in GPa.
z	Propagation distance over which correction is applied; positive values indicate the direction of wave travel, in meters (m).
Output Variable	**Description**
x_strain	Strain signal after applying dispersion correction.
x_stress	Stress signal after dispersion correction, expressed in MPa.

To launch the algorithm, the following command-line instruction must be issued:

`dispersion(x,fs,a,c0,E,z)`

Internally, this subroutine adheres to the correction philosophy proposed by Tyas and Pope [10], ensuring that the inferred axial stresses and strains accurately reflect the specimen's actual response.

The logic flow executed by the function can be distilled into the steps below:

1. Transformation of the incoming strain history into its frequency-domain counterpart is carried out using the FFT.
2. An ideal low-pass filter is engaged to strip away any spectral content that exceeds the upper bound dictated by the $M1$ correction.
3. For each frequency component that survives below the Nyquist limit, the following set of operations is performed:
   a. The script `disperion_factors.py` is queried to supply both the necessary phase offset and the two multipliers $M1$ and $M2$. Rather than performing expensive calculations on the fly, the procedure draws on a normalized, precomputed lookup table generated beforehand by `phase_velocity.py`, markedly reducing execution time.
   b. Leveraging the amplitude multiplier $M1$ in combination with the phase correction term $\theta'\omega$, a dispersion-compensated strain component is reassembled via the complex exponential formulation of the Fourier series previously shown as Eq. 7 and restated below as Eq. 8:

$$z_\varepsilon = M_1 A e^{I(\theta - \theta'_\omega)} \tag{8}$$

where A represents the raw amplitude of that spectral line, and θ its raw phase angle.

(c) An analogous reconstruction is performed for a dispersion-compensated stress component, this time enlisting both the strain-related multiplier $M1$ and the modulus-related multiplier $M2$ alongside the phase offset $\theta'\omega$, as formalized in Eq. 9 immediately following:

$$z_\sigma = M_1 M_2 A e^{I(\theta - \theta'_\omega)} \tag{9}$$

4. The portion of the spectrum residing above the Nyquist frequency is filled out by taking the complex conjugates of the corrected stress and strain vectors computed in the lower half.

5. Once assembly of the frequency-domain representations is complete, the numpy library's inverse FFT routine () is employed to convert these back into the time domain. The resulting waveforms are handed back to the caller through the output designators `x_strain` and `x_stress`.

These updated bar stresses and strains are then utilized within the broader `SHPB_Processing.py` framework to interpret the mechanical response of the material specimen under test.

Operation of dispersion_factors.py

Serving as a dependency for the larger `dispersion.py` program, the Python utility `dispersion_factors.py` provides the numerical coefficients necessary for dispersion adjustment. The core dispersion data used by this script originates from the separate utility

`phase_velocity.py` and reflect a Poisson's ratio of 0.29, a value consistent with the physical Hopkinson bars deployed in the accompanying experimental campaign.

Upon being called, `dispersion_factors.py` first loads four preparatory data files—designated `m1`, `m2`, `norm_freqs`, and `v_ratios`—and then proceeds to determine the amplitude and phase adjustments needed to compensate for dispersion at whichever discrete frequency has been requested.

The invocation that sets the script into motion is shown below, while the meaning of each argument and returned variable is elaborated in **Table 3**:

```
dispersion_factors(f,a,c0,z)
```

Table 3. Summary of arguments passed into and returned from `dispersion_factors.py`.

Parameter	Explanation
f	Signal frequency, expressed in Hz.
a	Radius of the bar, measured in meters (m).
c_0	Longitudinal wave speed in the bar (one-dimensional), given in m/s.
z	Propagation distance over which the correction is applied, in meters (m).
Output Variable	Explanation
angle_mod	Phase angle adjustment applied during correction, expressed in radians (rad).
m1	Adjustment factor accounting for non-uniform response across the bar's cross-sectional area.
m2	Correction factor representing variations in the relationship between axial stress and axial strain (i.e., dynamic Young's modulus).

The modified phase quantity `angle_mod` together with the two factors `m1` and `m2` are subsequently handed across to `dispersion.py`, so the appropriate signal phase translation can be performed and the revised strain and stress records obtained. The conceptual starting point for this module was a MATLAB routine originally written by Barr [19].

Operation of phase_velocity.py

Available as a self-contained open-source tool on both GitHub and ORDA [20], `Phase_velocity.py` has been built with a single objective: solving the equation proposed by Bancroft [6] for its first root via the bisection method. The calculation proceeds for a chosen Poisson's ratio and across a span of normalized wavelengths (d/L) specified by the user. The outcome is the normalized phase velocity $cp/c0$ tied to the lowest-order longitudinal propagation mode in an elastic cylindrical bar.

A conversion step maps the normalized wavelengths into normalized frequencies of the form $fa/c0$. With normalized phase velocities in hand, the code proceeds to determine the multipliers $M1$ and $M2$ introduced by Tyas and Watson [11], which serve to quantify how strain and Young's modulus, respectively, vary across the bar's radius in a wavelength-dependent manner.

The invocation that launches the script is shown below, accompanied by **Table 4**, which explains each input and output quantity:

```
phase_velocity(nu,l_ratios)
```

Table 4. Arguments supplied to, and values returned by, `phase_velocity.py`.

Parameter	Explanation
v (nu)	Poisson's ratio of the bar material utilized in SHPB experiments.
l_ratios	Normalized wavelength interval used to determine the first root of the Bancroft [6] equation.
Output variable	Explanation
dispersion_factors	A directory containing four pickle (.pkl) files that store dispersion-related parameters, including m1, m2, norm_freqs, and v_ratios.

From this point, the four outputs—`m1`, `m2`, `norm_freqs`, and `v_ratios`—feed into `dispersion_factors.py` directly, and `dispersion.py` by extension, so that dispersion compensation of the recorded SHPB waveforms can be executed as part of the parent routine `SHPB_Processing.py`. The design draws its inspiration from a MATLAB codebase created by Barr [21].

Practical applications

SHPB testing

The experimental work used a split-Hopkinson pressure bar assembly comprising two stainless-steel bars—the incident side and the transmitter side—arranged as depicted in **Figure 11**. Strain gauge stations essential for data extraction were placed such that the measurement point on the incident bar sat 1000 mm away from the specimen's leading edge, while the corresponding point on the transmitter bar was positioned 500 mm behind the specimen's trailing edge. At each station, a pair of Kyowa KSP-2-120-E4 semiconductor gauges captured the wave signatures.

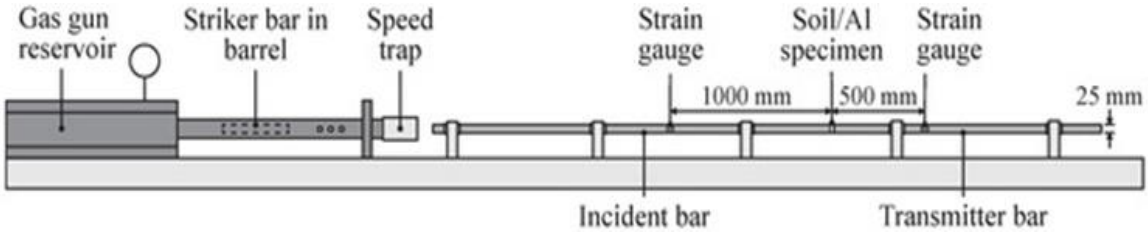


Figure 11. Diagrammatic representation of the SHPB arrangement used for the experiments.

Kaolin clay

An unconfined SHPB experiment was performed on a 25 mm-diameter kaolin clay specimen using the setup described in Section 5.1. The specimen's starting length was 5.357 mm, its overall mass 4.466 g, and its dry mass 3.167 g. Unprocessed voltage histories belonging to the incident and transmitter bars arrived via channels 7 and 8. Gain factors of 10 and 100 were applied to the incident and transmitter bar signals, respectively. The dispersion correction pathway was engaged. Waveforms were synchronized at their onset, and the striker bar's speed was measured. Data reduction proceeded through `SHPB_Processing.py`, called as follows:

```
SHPB_Processing(csvfile,[5.357,4.466,3.167],
'None',[7,8,5],
        [10,100,1],True,'start',True)
```

Sand

A confined SHPB trial was conducted on a 25 mm diameter specimen of medium sand, again using the arrangement described in Section 5.1. The specimen's initial length was 4.726 mm, and its total and dry masses were 3.50 g. Raw signals from the incident and transmitter bars were captured on channels 1 and 2, while the confining ring's response was routed through channel 3. Amplification was set to 10 on both bar channels. Dispersion correction remained active. Alignment was performed at the waveform start, and the gas-gun-driven striker bar's velocity was recorded. The invocation of `SHPB_Processing.py` took the form below:

```
SHPB_Processing(csvfile,[4.726,3.50,3.50],'R
ing',[1,2,3],
        [10,10,1],True,'start',True)
```

Aluminum

An aluminum specimen, 12 mm in diameter, was loaded without confinement in the SHPB configuration presented in Section 5.1. Its initial length was 5.000 mm, while the total and dry masses were each 1.530 g. Raw incident bar data were logged on channels 1 and 2. Both the incident and transmitter bar signals received a unity amplification factor of 1. Dispersion correction was switched on. The traces were aligned at the start, and on this occasion, no measurement of the striker bar's velocity was taken. Processing called upon `SHPB_Processing.py` with the command reproduced here:

```
SHPB_Processing(csvfile,[5.000,1.530,1.530],
'None',[1,2],
        [1,1],True,'start',True)
```

Comparative analysis of the SHPB-tested scenarios

With all three SHPB experiments put through the `SHPB_Processing.py` workflow, the tool's versatility is clearly demonstrated, as materials of sharply contrasting characteristics were each processed without difficulty. In every case, the act of dispersion correction effectively adds 1000 mm of propagation to the incident wave. It removes 1000 mm of propagation from the reflected wave while preserving the frequency-dependent phase velocity discussed in earlier sections.

Deriving the stress at the specimen's leading face requires summing the incident and reflected pulses; as a result, the reliability of the derived front stress is substantially improved. The contrast between dispersion-corrected outcomes and those obtained through basic time shifting is laid bare in **Figures 12-14**.

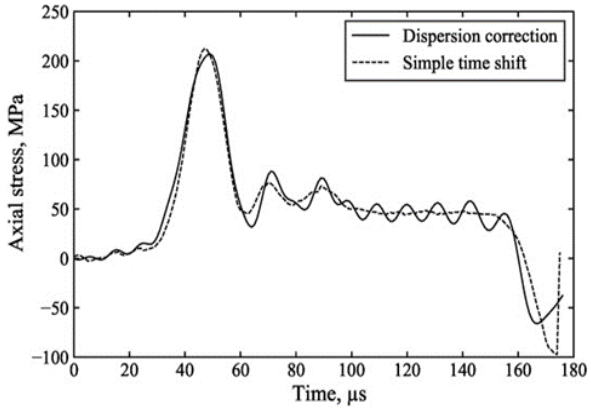
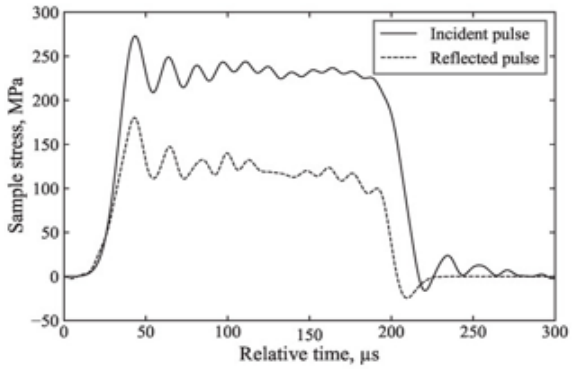
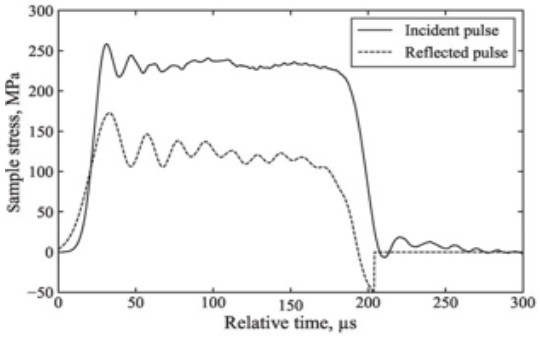


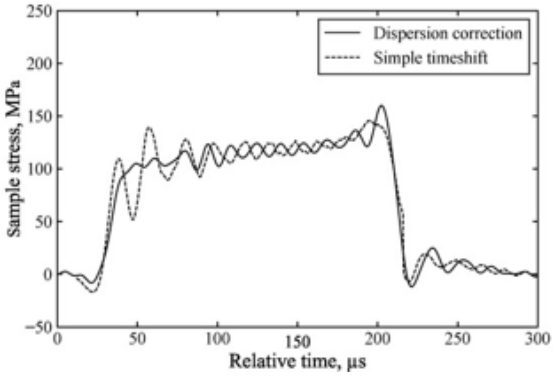
Figure 12. Front stress comparison—dispersion correction against time shifting—for the unconfined kaolin clay SHPB experiment.



a)



b)



c)

Figure 13. Output from the unconfined aluminum SHPB experiment: (a) incident and reflected pulses after dispersion correction, (b) the same pair of pulses following a simple time shift, and (c) a direct overlay of the resulting front stress from both approaches.

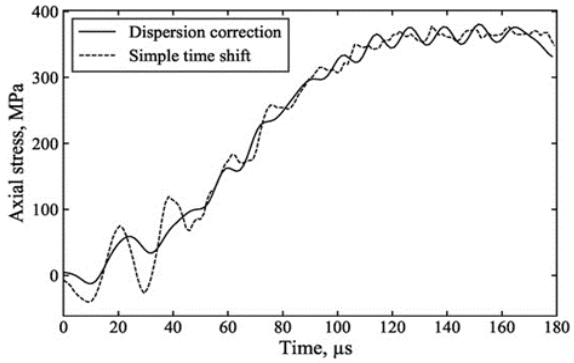
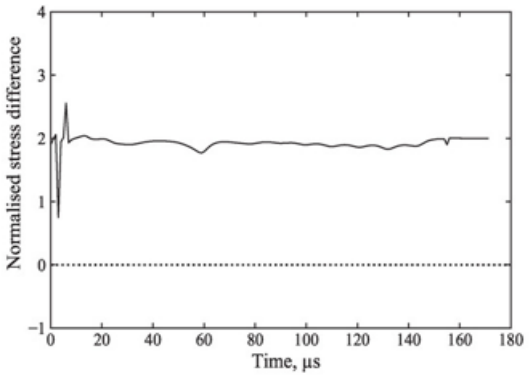


Figure 14. Front stress comparison—dispersion correction against time shifting—for the confined sand SHPB experiment.

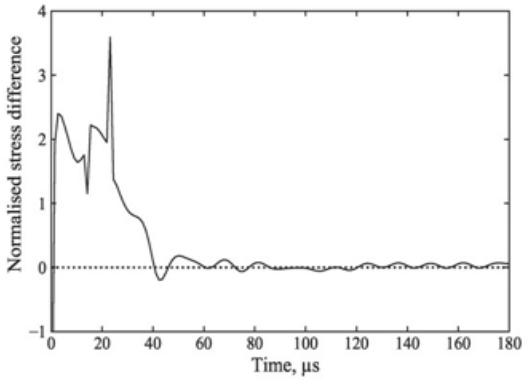
Picking out an illustrative case from **Figure 13**, the raw incident and reflected pulses are marred by pronounced oscillations in their early portions, an artifact of dispersion accompanying the steep stress onset. With dispersion correction active **(Figure 13a)**, the 1000 mm propagation leg from the gauge position to the specimen is explicitly compensated, so the morphing shape of those oscillations is correctly handled, leading them to mutually annul when specimen front stress is evaluated **(Figure 13c)**. When nothing more than a rigid temporal translation is imposed **(Figure 13b)**, those alterations in shape and relative positioning remain completely unaddressed. Consequently, the back-calculated front stress is contaminated with fictitious oscillatory features. The comparison underscores that dispersion correction supplies axial stress data of markedly higher fidelity and a considerably less ambiguous view of specimen response. The extra computation time incurred by the dispersion correction is trivial, on the order of 5 s.

Stress wave equilibrium

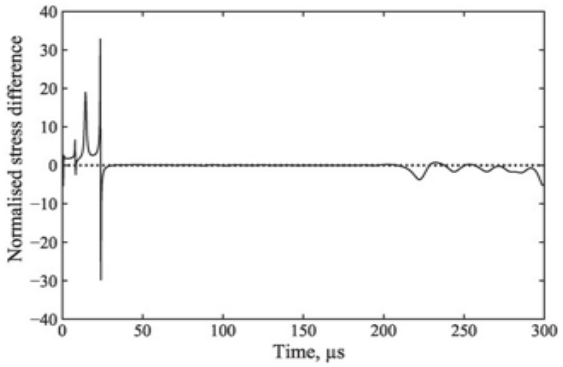
Charts were produced for all three cases, showing the mismatch between the front and back stresses, normalized by their mean values, as displayed in **Figures 15a–15c**. The purpose was to confirm that `SHPB_Processing.py` executes without failure and delivers dependable outcomes, irrespective of whether the stress waves have reached a state of equilibrium.



a)



b)



c)

Figure 15. Normalized front-to-back stress difference plotted for (a) an unconfined SHPB trial on kaolin clay, (b) a confined SHPB trial on medium sand, and (c) an unconfined SHPB trial on aluminum.

In a perfect scenario where stress equilibrium is fully realized during an SHPB experiment, the pulses sensed at the two bar–specimen boundaries—one at the front, one at the back—will match in duration. Reality, however, often deviates from this ideal, particularly when the stress wave fails to propagate throughout the entire specimen, leaving a significant portion to travel laterally.

Eq. 10 captures the condition of stress equilibrium during SHPB loading, provided the specimen undergoes uniform deformation, and the axial advance of the stress wave has been properly accounted for:

$$\varepsilon_f(t) = \varepsilon_r(t) + \varepsilon_t(t) \tag{10}$$

Owing to the alignment control built into `SHPB_Processing.py` for coordinating the front and back stress records, the code can accommodate situations where equilibrium is never quite established, yet still return a reasonable axial stress approximation. A note of caution is warranted, however: such estimates should always be supplemented with additional experimental work or numerical modeling before being adopted for formal material characterization.

Results and Discussion

The master routine that handles the reduction of raw SHPB voltage traces is `SHPB_Processing.py`. Embedded inside it sits `dispersion.py`, a sub-program charged with applying dispersion adjustments to the experimental waveforms. That sub-program in turn draws upon yet another utility called `dispersion_factors.py`.

What `dispersion_factors.py` does reshape the correction coefficients that `phase_velocity.py` has previously evaluated. Those coefficients are governed by the Poisson's ratio of whichever cylindrical bar forms part of the SHPB rig, and they can be generated with little effort for any bar material. Both this utility and the subordinate scripts it relies upon are fully self-contained and can be operated separately.

As the descriptive section of this article reveals, the tool packs an extensive set of capabilities: confinement handling, signal gain specification, a choice between dispersion correction and plain time alignment, waveform synchronization, striker velocity logging, real-time test log oversight, and automatic data archiving. On top of this, because the input and output signals belonging to the Hopkinson bars are all treated independently, the script runs smoothly regardless of whether stress wave equilibrium is ever attained. Staying true to its name, the code focuses on SHPB data reduction, thereby making the entire procedure considerably more time-efficient.

Real-world performance of the script was gauged through SHPB experiments on aluminum, kaolin clay, and sand. An unconfined aluminum specimen, an unconfined kaolin clay specimen, and a confined medium sand specimen were each subjected to loading in an SHPB apparatus. The majority of the script's feature set was engaged during the analysis of these trials; most critically, `dispersion.py` was employed to place dispersion-corrected and time-shifted results side by side, bringing into sharp relief how essential the script is for trustworthy data interpretation.

The current study has shown, through the practical deployment of `SHPB_Processing.py` on SHPB datasets gathered from alfrom aluminum, kaolin clay, and sand specimens, that the tool offers a compelling combination of speed, precision, and versatility across a wide range of applications.

Two of the three confinement modes supported by the algorithm—confined and unconfined SHPB configurations—were used in the practical applications segment. Trialing the code on an SHPB experiment that makes use of a partial lateral confinement arrangement would be of considerable worth for assessing processing fidelity. One of the program's genuine strengths is its ability to deliver across a range of testing regimes, with or without achieving stress-wave equilibrium during the SHPB event.

The script was executed within an SHPB configuration using stainless-steel pressure bars with a Poisson's ratio of 0.29. That said, evaluating the script's behavior on SHPB setups built from aluminum or polymer bars—materials that likewise call for dispersion compensation—would offer considerable additional insight.

Conclusion

A thorough exploration of the foundational principles underpinning dispersion correction and its significance for SHPB experimentation was undertaken. In response, a highly valuable computational asset was developed: `SHPB_Processing.py`, furnished with self-contained subroutines that augment the already wide-ranscript's

already wide-ranging feature set. The deployment of this tool on SHPB datasets from aluminum, kaolin clay, and sand specimens attests to a tangible improvement in result quality, underscoring the considerable promise this open-source algorithm holds for future uptake.

To conclude, this work highlights the indispensable contribution of split-Hopkinson pressure bar (SHPB) testing to advancing geotechnical research. By addressing wave dispersion and incorporating a rigorous correction scheme, the methodology put forward markedly improves the fidelity and reliability of stress evaluations in soils and other media. These outcomes reinforce the worth of SHPB testing as a lens through which dynamic material response can be understood, delivering essential knowledge for geotechnical endeavors.

Code availability

The algorithms developed in this paper are open-source and accessible on GitHub and ORDA at the following links:

- `gauge_factor.py`: GitHub and ORDA
- `phase_velocity.py`: GitHub and ORDA
- `SHPB_Processing.py`: GitHub and ORDA
- `dispersion.py&dispersion_factors.py`: GitHub and ORDA

Acknowledgments: None

Conflict of interest: None

Financial support: This research was funded by the Engineering and Physical Sciences Research Council (EPSRC) and the Defence Science and Technology Laboratory (DSTL).

Ethics statement: None

References

1. Shin H. Pochhammer–Chree equation solver for dispersion correction of elastic waves in a (split) Hopkinson bar. Proc Inst Mech Eng C J Mech Eng Sci. 2022;236(1):80–7.
2. Kolsky H. Radar. In: Stress waves in solids. New York (NY): Dover Publications Inc.; 1963. p. 140–5.
3. Shin H. Manual for calibrating sound speed and Poisson's ratio of (split) Hopkinson bar via dispersion correction using Excel® and Matlab® templates. Data. 2022;7(4):55.
4. Pochhammer L. On the propagation velocities of small oscillations in an infinite isotropic circular cylinder. J Reine Angew Math. 1876;81:324–36.
5. Chree C. The equations of an isotropic elastic solid in polar and cylindrical co-ordinates: their solution and application. Trans Camb Philos Soc. 1889;14:250–369.
6. Bancroft D. The velocity of longitudinal waves in cylindrical bars. Phys Rev. 1941;59(6):588.
7. Barr AD. Strain-rate effects in quartz sand [PhD thesis]. Sheffield (UK): University of Sheffield; 2016.
8. Gray GT III. Classic split-Hopkinson pressure bar testing. In: Kuhn H, Medlin D, editors. Mechanical testing and evaluation. Materials Park (OH): ASM International; 2000. p. 462–76.
9. Shin H. Sound speed and Poisson's ratio calibration of (split) Hopkinson bar via iterative dispersion correction of elastic wave. J Appl Mech. 2022;89(6):061007.
10. Tyas A, Pope DJ. Full correction of first-mode Pochhammer–Chree dispersion effects in experimental pressure bar signals. Meas Sci Technol. 2005;16(3):642–52.
11. Tyas A, Watson AJ. An investigation of frequency domain dispersion correction of pressure bar signals. Int J Impact Eng. 2001;25(1):87–101.
12. Gorham D. A numerical method for the correction of dispersion in pressure bar signals. J Phys E Sci Instrum. 1983;16(5):477–9.
13. Follansbee PS, Frantz C. Wave propagation in the split Hopkinson pressure bar. J Eng Mater Technol. 1983;105(1):61–6.
14. Barr A, Rigby S, Clayton M. Correction of higher mode Pochhammer–Chree dispersion in experimental blast loading measurements. Int J Impact Eng. 2020;139:103526.
15. Davies RM. A critical study of the Hopkinson pressure bar. Philos Trans R Soc Lond A Math Phys Sci. 1948;240(821):375–457.
16. Van Lerberghe A, Li KSO. Gauge_Factor.py: a Python algorithm for calculating the gauge factor of the input bar for split-Hopkinson pressure bar experiments. Sheffield (UK): University of Sheffield; 2023.
17. Li KSO, Van Lerberghe A, Barr A. SHPB_Processing.py: an open-source Python algorithm for correcting stress wave dispersion in split-Hopkinson pressure bar experiments. Sheffield (UK): University of Sheffield; 2023.
18. Van Lerberghe A, Barr A. Dispersion.py: a Python algorithm for phase angle and amplitude correction of pressure bar signals. Sheffield (UK): University of Sheffield; 2023.
19. Barr A. Dispersion.m: a MATLAB script for phase angle and amplitude correction of pressure bar signals. Sheffield (UK): University of Sheffield; 2016.
20. Van Lerberghe A, Barr A. Phase_Velocity.py: a Python algorithm for calculating frequency-

dependent phase velocity and radial variation of elastic waves in cylindrical bars. Sheffield (UK): University of Sheffield; 2023.
21. Barr A. Phasevelocity.m: a MATLAB script to calculate the frequency-dependent phase velocity and radial variation of elastic waves in cylindrical bars. Sheffield (UK): University of Sheffield; 2023.