# Efficiently downsampling inelastic cyclic stress-strain data while preserving its curvature

A Hartloper

July 26, 2021

**Abstract**

A downsampling algorithm for stress-strain data that uses few points but retains curvature within a pre-defined tolerance is developed in this document.

## 1 Introduction

The raw data (i.e., extensometer displacement, load cell force) in experiments used for material model calibration are typically sampled at 10 Hz in the EPFL test setup. As experiments range from around a few minutes (e.g., LP1) to hours (LP5), this results in thousands to hundreds of thousands of data points for each experiment. However, the sampled data is often denser than required to reasonably describe the stress-strain behavior of materials under the applied strain rates. When it comes to calibrating material models, the time required is proportional to the number of data points. Therefore, time can be saved by reducing the number of data points prior to running the calibration while maintaining the fidelity of the data.

Experience has shown that the fidelity of an experiment can be kept with around 10–100 times less points depending on the load protocol. Calibrating the UVC model with ten load protocols *without* reducing the number of points is, therefore, could take weeks to a year. A "good" method of reducing the number of data points, or *downsampling* the data, is essential to keep the calibration time reasonable (say, on the order of hours to days per material). The properties that define a good downsampling method in this context are now defined.

A good downsampler for inelastic, cyclic, stress-strain data of structural steels:

1. samples enough points in the initial elastic region to obtain a good estimate of the initial elastic modulus,

2. samples the upper yield point,

3. samples all the maxima and minima in each loading cycle (i.e., the *peaks* of the stress-strain graph),
4. samples enough points to retain the fidelity of regions in the stress-strain graph with high curvature,
5. samples as few points as possible in regions of low curvature,
6. can handle different load protocols,
7. can handle noise in the measurements,
8. can handle different strain rates,
9. has interpretable heuristic parameters,
10. is easy to use.

An algorithm that combines different techniques to satisfy these desirable properties is developed in this document.

My initial work to downsample stress-strain data combined two techniques: (i) find the peaks of the stress-strain data, then (ii) downsample by an integer factor[1]. This method preserves the peaks, however, it either removes too many points in the curved regions, or is inefficient in the straight regions. Moreover, it is a matter of "art" to pick a reasonable integer factor for each case, and, this has to be done twice for each experiment because of the different strain rates in most load protocols! There are several areas to improve this method that lead to this study.

The initial inspiration for improvement was this blog[2]. Then I found a wealth of literature, particularly in the domain of cartography. Two subproblems are defined in the literature: the min-# problem and the min-$\epsilon$ problem. The objective of the first problem is to determined the minimum number of sample points to provide a reasonably good approximation of the curve, the second is to find the minimum error between the original and sampled curves for a given number of points [ref imai+iri]. In the first, an error critieria is specified ($\epsilon$), and in the second the number of points is specified ($N$). Many approximate solutions exist [RDP + survey] as well as optimal ones [imai and chan+chin, perez+vidal] The optimal solution to the first problem is easier (takes less computational effort) than the second one by a factor of $\log N$ [ref chan chin]. I have focused on applications of approximate solutions to the min-# problem.

The most popular algorithm to determine an approximate solution to the min-# problem appears to be the Ramer–Douglas–Peucker (RDP) algorithm

[1] `https://en.wikipedia.org/wiki/Downsampling_(signal_processing)`

[2] `https://kaushikghose.wordpress.com/2017/11/25/adaptively-downsampling-a-curve/`.

[ref rdp + survey]. The RDP algorithm is not optimal and only finds a number of points $N$ close to the minimum. A number of additional points are certain to be added due to the criteria of selecting sample points. I.e., as the point at maximum perpendicular distance to the line between first and last points. This heuristic makes sense for graphs that are not "cyclic". However, the RDP algorithm is still quite good, comparatively fast, and has been widely implemented. Excellent visualizations of the RDP algorithm can be found[3].

# 2 Proposed procedure

The proposed algorithm is composed of three parts. The first part is to find the peaks in the stress-strain data. The second part is to downsample the entire stress-strain history based on the curvature of the stress-strain data. The third part is to keep additional points in the initial elastic region. The final downsampled data is the unique set combining all three parts.

Mosts of the heavy lifting is done in the second part. The first part is essential to overcome *aliasing* in the sampled data and determine properties such as the number of cycles. The third part is essential to ensure fidelity of the initial elastic region. Details are given for each part and a summary of the entire procedure is given afterwards.

## 2.1 Identifying peaks in the stress-strain graph

The peaks in the stress-strain graph are identified by the minima or maxima in each cycle. Cycles can be fairly reliably identified by the switching of signs in the stress signal. Therefore, the peaks are the minima and maxima between each change in sign of the stress signal. This method works well if the stress crosses zero in each cycle and there is no substantial noise in the stress signal around zero.

An attempt at identifying upper yield point, if it exists, is made in this first part. First, the 0.2 % offset yield stress is computed using the initial elastic modulus. The initial elastic modulus itself is computed using linear regression up to $0.65 f_{y,n}$, where $f_{y,n}$ is the nominal yield stress of the material. Then, the upper yield point is selected as the point of maximum stress up to the 0.2 % offset yield stress. If there is no upper yield point (due to the material or imperfections in the specimen), then the maximum will typically be at the 0.2 % offset yield stress, and this point is sampled instead.

Another important point in stress-strain data tested according to the RESSLab protocols is the first instance 2 % strain amplitude is crossed. This point is important because the strain rate changes by a factor of around 30 at this point. Therefore, this point is identified as well. The final point

[3] `https://karthaus.nl/rdp/`

of importance is the final point. Unless otherwise specified (i.e., due to buckling), the final point is selected at 12.5 % strain due to limitations in the small extensometer used for M8 experiments, or the last point in the history if 12.5 % strain is never obtained. Any points past the final point are neglected.

To summarize, the sampled points from the first part are:

- the upper yield point (or 0.2 % offset yield stress point),
- the point where 2 % strain amplitude is crossed (not applicable for LPs 4 and 5),
- the peaks in the stress-strain graph, and
- the final strain point.

## 2.2 Curvature-preserving downsampling

The idea in the second part of the overall procedure is to remove as many points as possible in the "straight" regions of the stress-strain graph and keep as many as necessary in the "curved" regions. The key to these methods is to define what is "straight", and to sample a point every time that this condition is violated.

### 2.2.1 First attempt: Maximum deivation downsampler

The method, named the "maximum deviation downsampler", is outlined in Algorithm 1 based on a blog by Kaushik Ghose in footnote 2. After further reading I realized the link with other methods outlined in the introduction. This method requires a set of points and a tolerance, $\epsilon$. The algorithm starts at the first point and steps through each point in the set. A line is computed between the starting point and the current point. The perpendicular distance is computed between the line and each point in the range of the starting and current point. If the perpendicular distance is greater than the tolerance, the point before the current point is sampled; the sampled point is then set as the starting point and the process continues.

The keys to making Algorithm 1 in the current context are to: compute an appropriate perpendicular distance, to use an appropriate tolerance, and to remove noise from the data. One challenge in stress-strain data is that the "distances" in stress and strain have different units. This is handled by normalizing the stress data by the range in stress ($\sigma_i^* = \sigma_i/(\max\sigma - \min\sigma)$), and the strain by the range in strain ($\varepsilon_i^* = \varepsilon_i/(\max\varepsilon - \min\varepsilon)$). Therefore, the normalized values have a range of 1.0 in both axes.

Given this normalization, a tolerance of $\epsilon = 0.001$ is usually appropriate. This tolerance can be interpreted that "straight" is defined as a deviation less than 1/1000'th of the maximum "distance" spanned by the graph. Finally,

noise in the data can cause spurious sampling of points. Therefore, a moving average filter (Savitzky-Golay, `scipy.signal.savgol_filter`) is applied to the stress-strain data prior to normalization. The moving average filter requires a window length, $w$ and an interpolation order, $p$. Filtering the stress-strain data removes the noise, but leads to aliasing. Recall that the peaks of the stress-strain signal are already included in part 1 of the overall procedure, therefore, aliasing is not an issue in part 2. Therefore, the set of 2-D points $\{x_i\}$ input to Algorithm 1 are the filtered, normalized stress-strain data.

**Algorithm 1** Maximum deviation downsampler.

**input:** Set of points, $\{x_i\}_{i=0}^{N-1}$ with $x_i \in \mathbb{R}^n$ and $N \in \mathbb{Z}$; and tolerance, $\epsilon \in \mathbb{R}$.
**output:** Set of indices between $[0, N)$ to keep, $k_{sample}$.
$k_{sample} \leftarrow \emptyset$
$i \leftarrow 0$
**for** $j \in (0, \ldots, N)$ **do**
    $l_{ij} \leftarrow x_j - x_i$ ▷ Line between $i$ and $j$
    **for** $k \in (i, \ldots, j)$ **do**
        $d_{ij}^{(k)} \leftarrow x_k \perp l_{ij}$ ▷ Perpendicular line between $x_k$ and $l_{ij}$
    **end for**
    **if** $\max \left\| d_{ij}^{(k)} \right\|_2 > \epsilon$ **then**
        $k_{sample} \leftarrow k_{sample} \cup \{j-1\}$
        $i \leftarrow j - 1$
    **end if**
**end for**

### 2.2.2 Second attempt: The RDP algorithm

The RDP algorithm is similar to the maximum deviation downsampler in that it selects sample points based on local perpendicular distances. What differs is the choice of starting sample points and how sample points are progressively chosen. The process is illustrated in Figure 1. The RDP algorithm starts with $i = 0$ and $j = N - 1$, then the first sample point is that with maximum $d_{ij}^{(k)}$. This bisects the overall curve, and each section is sampled recursively until the maximum deviation is lower than the tolerance in all sections. Notice that several points are "skipped" in the regions of low curvature, while the density of points sampled increases in the curved regions. The same need for normalization and sensitivity to noise exist in the RDP algorithm as in the maximum deviation downsampler. Therefore, the stress-signal is filtered prior then normalized by the ranges in stress/strain prior to applying the RDP algorithm.

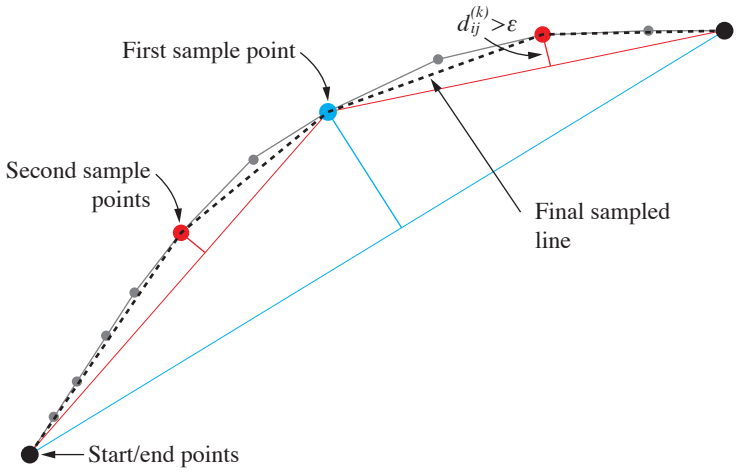



Figure 1: RDP algorithm process. Starts by sampling the end points, then recursively sampling the point with maximum perpendicular distance between each set of sample points. Stops when the maximum perpendicular distance is less than the tolerance.

### 2.2.3 A related approach: Limiting global error

Both the max deviation downsampler and the RDP algorithm operate using a local tolerance, $\epsilon$. However, it may be more appropriate to use a global tolerance on the difference between sampled and original data. The downsampled data can be compared with the original data using our "standard" error metric:

$$\bar{\varphi} = \sqrt{\frac{\int (\sigma_{test} - \sigma_{ds})^2 \mathrm{d}\varepsilon^*}{E_{orig}}}, \tag{1}$$

where $E_{orig}$ is the total accumulated squared strain energy in the original data, $\sigma_{test}, \sigma_{ds}$ are the stress signals of the original and downsampled data, and $\varepsilon^*$ is the accumulated strain. Linear interpolation is used to compute the downsampled data at each point between the sampled points, and the integrals are computed using the trapezoidal rule.

An algorithm to keep the error metric below a global tolerance, $\epsilon_g$ is provided in Algorithm 2. The idea is to select an initial ("large") local tolerance, $\epsilon$. Then reduce the local tolerance and re-run the downsampling until $\bar{\varphi} < \epsilon_g$. This always converges because the error is zero at the limit of sampling all the points. This procedure is made accessible by the highly efficient implementation of the RDP algorithm in the Python package `polyprox`, however, the same process can be applied using other downsampling methods with a bit of time. Note that my implementation uses the filtered/scaled "$\sigma_{test}$" to compute $\bar{\varphi}$, thus, the true error between the sampled data tends to be slightly higher.

**Algorithm 2** Downsampling based on global error.

**input:** Set of points, $\{x_i\}_{i=0}^{N-1}$ with $x_i \in \mathbb{R}^n$ and $N \in \mathbb{Z}$; initial local tolerance, $\epsilon_0 \in \mathbb{R}$; and global tolerance $\epsilon_g \in \mathbb{R}$.
**output:** Set of indices between $[0, N)$ to keep, $k_{sample}$.
$\chi \leftarrow 1.05$ ▷ Factor to help convergence when $\bar{\varphi}$ is near $\epsilon_g$
$n_g \leftarrow 10$ ▷ Maximum number of iterations
$i \leftarrow 0$
$\bar{\varphi} \leftarrow 10 \cdot \epsilon_g$ ▷ Just to be larger than $\epsilon_g$ initially
$\epsilon \leftarrow \epsilon_0$
**while** $\bar{\varphi} > \epsilon_g$ and $i < n_g$ **do**
    $k_{sample} \leftarrow$ indices from downsampling with local tolerance $\epsilon$
    $\bar{\varphi} \leftarrow$ error metric computed using $\{x_i\}$ and $k_{sample}$
    **if** $\bar{\varphi} > \epsilon_g$ **then**
        $\epsilon \leftarrow \epsilon \cdot (\epsilon_g / (\chi\, \bar{\varphi}))$
        $i \leftarrow i + 1$
    **end if**
**end while**

### 2.3 Additional sampling in the initial elastic region

The initial elastic region is assumed to be bounded by the first point in the data and the upper yield point obtained in part 1 of the overall procedure. It is important to have a few points in this region to obtain a good estimate of the initial elastic properties from the downsampled data. Therefore, $n_{elastic}$, additional points are sampled in the initial elastic region at approximately evenly spaced strain intervals. Each additional point is sampled using

$$j = \arg\min |x_i - x_t|, \tag{2}$$

where $x_t$ are the target evenly spaced strain points.

### 2.4 Saturation in constant amplitude tests

Cyclic hardening saturates logarithmically under constant amplitude cyclic loading in structural steel materials. Therefore, many cycles may be associated with a small increase in stress and a large portion of data is associated with a small amount of information. One option to reduce this data is to cut the cycles after saturation has been reached. The saturation point is defined herein as the index of the first peak $i$ that satisfies $\sigma_i > \eta \cdot \max \sigma$ in the positive loading direction and $\sigma_i < \eta \cdot \min \sigma$ in the negative loading direction, with $0 < \eta \leq 1$. The data after the saturation point is not sampled.

Cutting cycles after saturation in constant amplitude tests requires the definition of the saturation tolerance, $\eta$. The minimum number of cycles to include, $n_{cyc}$, is also specified.

### 2.5 Overall procedure

The overall procedure is summarized in Algorithm 3. The procedure consists of parts 1, 2, and 3 outlined earlier, with the incorporation of cycle cutting after saturation in constant amplitude tests. The original stress-strain data input to the procedure is $X = \{x_i\}_{i=0}^{N-1}$. Algorithm 3 produces a set of indices, $k_{sample}$. The downsampled stress-strain data is $X_{ds} = \{x_i \mid i \in k_{sample}\}$. I emphasize that the downsampled data is sampled directly from the original data, therefore, $X_{ds} \subset X$. (Just to emphasize that the filtered, normalized data is only used to determine the indices and the added elastic points are the closest indices to the targets.) The global error method can be used instead by swapping line 16 with the results from Algorithm 2.

I recommend using the following input parameters in Algorithm 3:

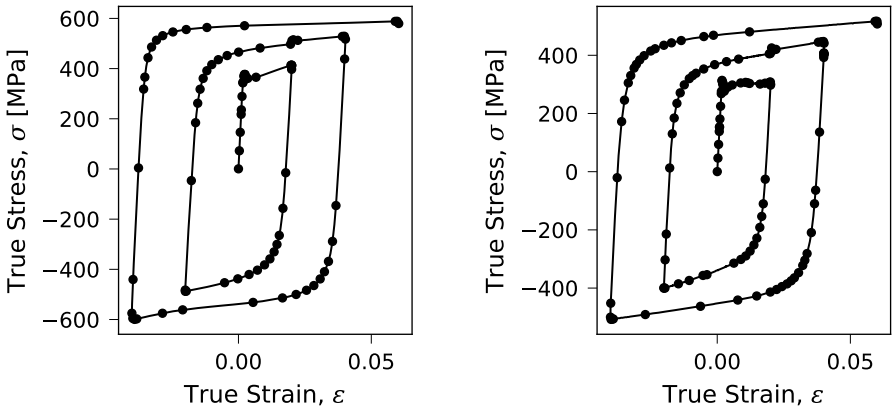
- $f_{yn}$: based on the material
- $\eta = 0.99$
- $n_{cyc} = 20$
- if local method: $\epsilon = 0.001$
- if global method: $\epsilon_0 = 0.1$, $\epsilon_g = 0.005$
- $w = 5$
- $\alpha = 11$
- $p = 1$
- $n_{elastic} = 7$

The parameter $\eta = 0.99$ was selected as being close to 1.0 and $n_{cyc} = 20$ was selected from [ref albano] because at least 20 cycles are indicated. Two window lengths are required for pre- and post-2 % strain due to the different strain rates. A value of $w = 5$ provides a weighted average of $\pm 2$ points, and $\alpha = 11$ was selected because, even though the strain rate is around 26 times lower, the window length lead to too much aliasing in the pre-2 % region with larger values. Linear interpolation ($p = 1$) is recommended because it is most effective at reducing the noise. The trade-off is that linear interpolation increase aliasing around peaks, however, the peaks are already included in part 1. Seven additional points are added to the initial elastic region to give nine points total (seven + start + upper yield). This should give 4–5 points in the $0.65 f_{yn}$ region used to compute the initial elastic modulus. The basis for $\epsilon$ recommendations will be shown through later examples.

**Algorithm 3** Overall summary of the proposed downsampling method.

**input data:** set of stress-strain data, $\{x_i\}_{i=0}^{N-1}$
**input parameters:** nominal yield stress, $f_{yn}$; saturation tolerance in constant amplitude loading, $\eta$; minimum cycles in constant amplitude loading, $n_{cyc}$; downsampler tolerance(s), $\epsilon$ or $\epsilon_0, \epsilon_g$; filter window length post-2 % strain, $w$; filter window length pre-2 % factor, $\alpha$; filter interpolation order, $p$; number of additional points in the initial elastic region, $n_{elastic}$.
**output:** $k_{sample}$ the indices of the sampled points.

Compute the initial elastic modulus and initial 0.2 % offset yield stress using the nominal yield stress, $f_{yn}$
$k_1 \leftarrow$ indices of the peaks in the stress-strain graph
**if** constant amplitude loading **then**
    Determine $n_{sat}$ cycles to reach $\sigma_i > \eta \cdot \max \sigma$ and $\sigma_i < \eta \cdot \min \sigma$
    **if** $n_{sat} < n_{cyc}$ **then**
        $n_{sat} \leftarrow n_{cyc}$
    **end if**
    Truncate data past $n_{sat}$ cycles
**end if**
Filter stress data pre-2 % strain with window length $\alpha \cdot w$ and order $p$
Filter stress data post-2 % strain with window length $w$ and order $p$
Normalize the stress-strain data
$k_2 \leftarrow$ indices from downsampler with local ($\epsilon$) or global method ($\epsilon_g, \epsilon_0$)
$k_3 \leftarrow$ indices of $n_{elastic}$ evenly spaced points in the initial elastic region
$k_{sample} \leftarrow k_1 \cup k_2 \cup k_3$




(a) HEM320 web dataset

(b) S235/275 15 mm plate dataset

Figure 2: Sampled points in LP8 from datasets. Line = test data, dots = sampled points.

# 3 Results

The results shown in this section were computed using the recommended parameters. Typical results are shown in Figure 2a.

Savings vs original data S235/275 Plate 15 75.826747 HEM320 web 90.042243 Savings vs old method S235/275 Plate 15 1.409398 HEM320 web 2.897669

# 4 Discussion

## 4.1 Choice of algorithm

The choice between the maximum deviation downsampler and RDP algorithms is discussed in this section. The max deviation downsampler is not recommended for two reasons: (i) the implementation is not optimal in that it does not guarantee the maximum perpendicular distance is always less than the minimum, and (ii) my implementation in Python is slow compared to the RDP algorithm in `polyprox`. The first issue being more critical, of course. The first issue is rooted in the nature of our data having many inflection points. With multiple inflection points, although $d_{ij}^{(k)} < \epsilon$, there may be some $d_{mj}^{(l)} > \epsilon$ with $m > i$ and $m < l < j$. Therefore, "stepping forward" as in the maximum deviation downsampler is not a good strategy to ensure that the maximum perpendicular distance is less than $\epsilon$. This makes the second issue not worth solving.

With regards to the RDP algorithm, the cyclic nature of our stress-strain

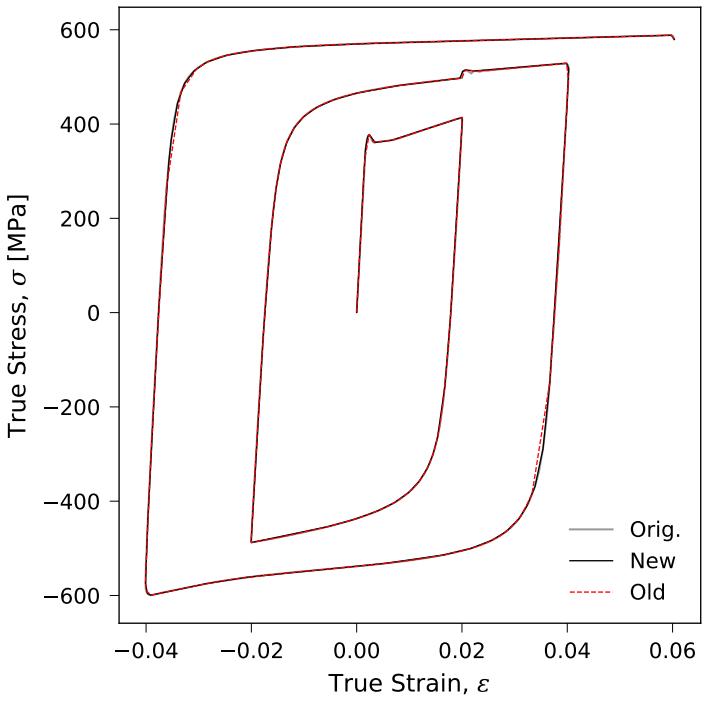
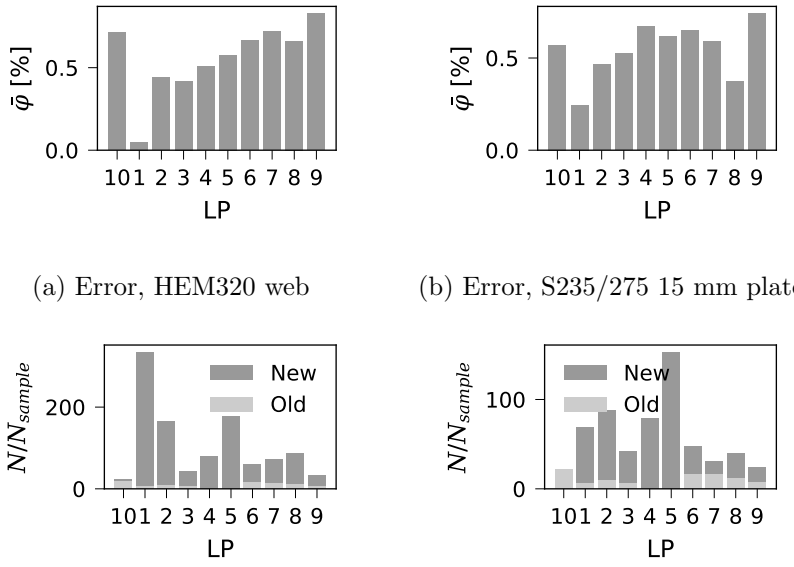



Figure 3: HEM320 web dataset

Figure 4: Comparison of proposed (New) and integer reduction (Old) methods of downsampling.




(a) Error, HEM320 web

(b) Error, S235/275 15 mm plate

(c) Reduction, HEM320 web

(d) Reduction, S235/275 15 mm plate

Figure 5: Error and Reduction factors between original and downsampled data. New is the proposed method, Old is the integer factor downsampling.

data implies that selecting the start and end points, then sampling the maximum perpendicular distance between these, is not a great heuristic. See Figure 1. This is because the maximum perpendicular distance from the line linking the start and end points is not necessarily at a region of high curvature. However, the RDP algorithm is not optimal in producing the minimum number of points, but it will be on the "safe side" of adding a few additional points to get the job done. The results could be more efficient (i.e., satisfy the local $\epsilon$ criteria with less points) by using an optimal solution from [ref chan chin].

### 4.2 Different distance measures

As outlined, a challenge in the max deviation downsampler is the difference in units between stress and strain. I propose to re-scale the stress-strain data to then use a basic 2-norm when computing the perpendicular distance. Two options that could be used rather than rescaling the data are:

1. use the angle between the line and each point,
2. use the norm induced by an alternative inner product: $\|d_k\|_S = \sqrt{d_k \cdot S \cdot d_k}$, and
3. use an energy criterion.

The first option is similar to the maximum perpendicular distance, but more sensitive to noise. This is because of the small angle rule, $\sin\theta \approx \theta$, and the perpendicular distance is $\|v\|_2 \sin\theta$, where $v = x_k - x_i$. If $\|v\|_2$ is small, the angle can have a large variation (i.e., in the case of high frequency noise), however, the perpendicular distance remains small.

In the second option, Algorithm 1 remains essential the same except for how $d_k$ is computed. To use a matrix-vector notation:

$$d_k = \begin{bmatrix} d_{\varepsilon,k} \\ d_{\sigma,k} \end{bmatrix}, \quad S = \begin{bmatrix} \frac{1}{(\max\varepsilon - \min\varepsilon)^2} & 0 \\ 0 & \frac{1}{(\max\sigma - \min\sigma)^2} \end{bmatrix},$$

where the values of $S$ would in this case give a similar effect to the scaling that I already use. Other options for $S$ could be selected. Essentially, we compute a scaled distance rather than scale the data itself. This method would still require a selection of tolerance $\epsilon$, but it may be different depending on the matrix $S$.

In the third option, the perpendicular distance criterion is replaced by the energy bounded by the polygon. In this case, instead of $d_k$'s we compute

$$e = \int_{\varepsilon_i}^{\varepsilon_j} (\sigma - l_\sigma) \mathrm{d}\varepsilon,$$

where $l_\sigma$ represents the stress signal of the line between $x_i$ and $x_j$. If the stress-strain graph is nearly straight, then $(\sigma - l_\sigma) \approx 0$. The point $j-1$ would be sampled if $e > \epsilon$.

The benefits of the second method are that it does not require any scaling of the data and the error is closely related to the calibration problem. However, the link between the energy, $e$, and the curvature in the stress-strain graph is less clear, making it more difficult to reason about an appropriate tolerance. Everything considered, a tolerance still needs to be selected regardless of the method. I went with the data scaling option because, in my opinion, the link between the algorithm and the geometric interpretation of curvature is most clear. Furthermore, this makes the overall procedure more easily adapted to existing algorithms (e.g., `polyprox`).

### 4.3 More than two dimensions

This document assumes two-dimensional data parametrized as a function of time, i.e., $x(t) \in \mathcal{D} \times \mathcal{T}$, where $\mathcal{D} \subset \mathbb{R}^n$ and $n$ is the dimension of the data ($n = 2$ in our case—uniaxial stress/strain), and $t \in \mathcal{T}$ is the time. Algorithm 3 can easily be extended to $n > 2$ by preserving curvature in each considered data pair. The same procedure can then be used with an appropriate tolerance, $\epsilon$, and an appropriate scaling procedure by sampling a point anytime the tolerance is exceeded by one of the pairs. Considering higher dimensions may be useful, for instance:

- when downsampling temperature and stress-strain data together (can preserve curvature in both temperature-strain and stress-stress), and
- when downsampling data considering multiple components of the stress-strain tensor (e.g., multiaxial material tests).

# 5 Conclusion