PROJECT REPORT :

Data-driven Computational Mechanics: Implementation and Application

Yannick Neypatraiky

Supervised by

Sacha Zenon Wattel

Prof. Jean-François Molinari

19th January 2023

Contents

1 Introduction

1.1 Context

Computational mechanics is a field that uses numerical simulations to solve complex mechanical problems. Traditionally, these simulations have relied on models, known as constitutive laws, to describe the behaviour of materials. These models attempt to match experimental results, but they can be limited by the assumptions made. In recent years, a new approach known as Data-Driven Computational Mechanics (DDCM) has emerged, which aims to use experimental data directly to solve problems, bypassing the modeling step and reducing the potential for bias.

DDCM differs from traditional constitutive model-based simulation in the sense that it relies on empirical knowledge. This allows the use of a much wider range of data and makes the method less sensitive to any constitutive modelling assumptions. Additionally, it also has the potential to overcome some of the limitations of traditional constitutive modelling, such as the difficulty of modelling complex nonlinear behaviour or the lack of experimental data to validate models. As such, it can be particularly useful in cases where there is limited understanding of the underlying physical processes, or where the behaviour of a material changes significantly with loading conditions.

1.2 Description

In this project, the data-driven computational mechanics approach is implemented and applied to analyse structural mechanics. The project is based on Kirchdoerfer and Ortiz article [1], where this new paradigm of data-driven computing was first developed. According to this paradigm, calculations are performed directly from experimental material data, pertinent constraints and conservation laws, bypassing the traditional step of empirical material modelling. The DDCM method implemented here is based on the idea of finding the closest state from a prespecified dataset to satisfying the conservation laws, and is equivalent to finding the state that satisfies the conservation laws that is closest to the dataset.

The project involves the development of data-driven solvers in Python, and their application to various examples, such as the static (or dynamic) equilibrium of a truss. The performance of the data-driven solvers is evaluated in terms of their convergence properties with respect to the number of data points, and with regard to local data assignment. The project aims to investigate the potential advantages and limitations of this approach compared to traditional methods and the robustness of data-driven solvers with respect to spatial discretization. Finally, the expected results of this project is that the data-driven solutions converge to the classical solution as the dataset approximates increasingly closely a classical material law in phase space, as highlighted in the reference paper [1].

2 . Data-Driven Solver

2.1 Truss structures

2.1.1 Theoretical principles

The Data-Driven scheme is introduced by considering a simple linear elastic truss problem. Truss structures are chosen here as they are composed of articulated bars that can only experience uniaxial stress and strain, making the material behaviour of a bar relatively simple to understand. Therefore, the use of trusses allows for a clear illustration of the underlying principles of the DDCM approach.

In linear elasticity, the behaviour of a truss bar follows the well-known Hook's law:

$$\sigma = E \cdot \epsilon \tag{2.1}$$

where ϵ is the strain, σ the stress and E the Young Modulus of the truss bar.

In the Data-Driven approach however, it is assumed that the material behaviour for each bar $e \in \{1, ..., m\}$ of the truss is represented by a set E_e, composed of different pairs (ϵ_e, σ_e), known as local states. These local states could be experimental measurements, subgrid multiscale calculations, or other data that can characterises material behaviour. It can be noted that this approach completely removes the necessity to determine the material properties such as the Young Modulus.

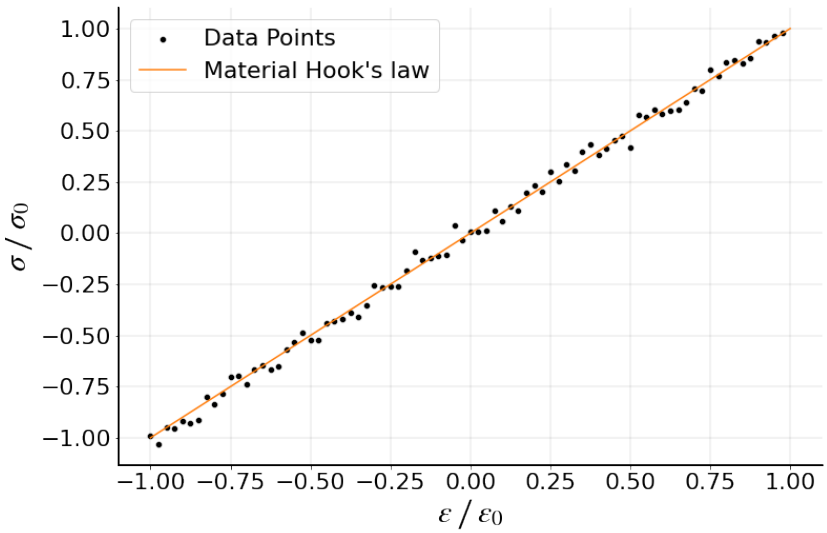

Figure 2.1: Example of normalised material dataset for a linear elastic truss bar

The entire space of pairs (ϵ, σ) is referred to as the phase space. Unlike traditional computational mechanics solvers, DDCM seeks to assign the best possible local state (ϵ_e, σ_e) from the corresponding dataset E_e to each bar e, while simultaneously satisfying compatibility and equilibrium. This leads to the concept of optimality of the local state, which is determined by minimising the distance between the data points and the space defined by fundamental laws, compatibility and equilibrium, within the phase space, using an appropriate penalty function.

The local penalty functions considered here are defined as:

$$\boldsymbol{F_e}(\epsilon_e, \sigma_e) = \min_{(\epsilon'_e, \sigma'_e) \in E_e} \boldsymbol{W_e}(\epsilon_e - \epsilon'_e) + \boldsymbol{W_e^*}(\sigma_e - \sigma'_e) \tag{2.2}$$

for each bar $e \in \{1, ..., m\}$ in the truss with,

$$\boldsymbol{W_e}(\epsilon_e) = \frac{1}{2} C_e \epsilon_e^2 \quad \text{and} \quad \boldsymbol{W_e^*}(\sigma_e) = \frac{1}{2} C_e \sigma_e^2 \tag{2.3}$$

with the minimum taken over all local states (ϵ'_e, σ'_e) in the local dataset E_e. The functions $\boldsymbol{W_e}$ and $\boldsymbol{W_e^*}$ are introduced as part of the numerical scheme and does not need to represent any actual material behaviour, but can be seen as reference strain and complementary energy densities, respectively. The constant C_e is also numerical and does not represent a material property.

In the same fashion as traditional methods, the global state of the truss in DDCM is obtained by combining all local states (ϵ_e, σ_e) of each one of its bars, leading to the formulation of the global penalty function:

$$\boldsymbol{F} = \sum_{e=1}^{m} w_e \cdot \boldsymbol{F_e}(\epsilon_e, \sigma_e) \tag{2.4}$$

with $w_e = A_e \cdot L_e$ denoting the volume of truss member e, A_e being its cross-sectional area and L_e its length.

2.1.2 Optimization problem

The aim of the data-driven solver being to find the optimal local states for each bar e. Mathematically it can be seen as minimisation problem where the global penalty function $\boldsymbol{F}$ is the cost whereas equilibrium and compatibility equations form the set of constraints.

The constrained minimisation problem can therefore be expressed as :

$$\begin{aligned} \text{Minimise:} \quad & \sum_{e=1}^{m} w_e \cdot \boldsymbol{F_e}(\epsilon_e, \sigma_e) \\ \text{subject to:} \quad & \epsilon_e = \sum_{i=1}^{n} B_{ei} \cdot u_i, \quad \forall \text{ bar } e \in \{1, ..., m\} \\ & \sum_{e=1}^{m} w_e \cdot B_{ei} \cdot \sigma_e = f_i, \quad \forall \text{ node } i \in \{1, ..., n\} \end{aligned} \tag{2.5}$$

where $\boldsymbol{u} = [u_1, ..., u_n]^T$ is the array of nodal displacements, $\boldsymbol{f} = [f_1, ..., f_n]^T$ is the array of applied nodal forces and the matrix $\boldsymbol{B}$, of size $m \times n$, encodes the connectivity and geometry of the truss.

Expressing the strains in terms of displacements allows to highlight the compatibility constraints while the equilibrium constraint can be enforced through the use of Lagrange multipliers, thus resulting in a stationary problem (see Eq.5 in [1]).

After solving the stationary problem and determining all optimal data points $(\epsilon_e^*, \sigma_e^*)$, a system of linear equations can be established for nodal displacements u_i, local stresses σ_e, and Lagrange

multipliers η_i. This system can be simplified and expressed in an equivalent form:

$$\sum_{j=1}^{n}\left(\sum_{e=1}^{m} w_e C_e B_{ej} B_{ei}\right) u_j = \sum_{e=1}^{m} w_e C_e \epsilon_e^* B_{ei} \tag{2.6}$$

$$\sum_{j=1}^{n}\left(\sum_{e=1}^{m} w_e C_e B_{ei} B_{ej}\right) \eta_j = f_i - \sum_{e=1}^{m} w_e B_{ei} \sigma_e^* \tag{2.7}$$

This system is composed of two standard linear-elastic truss-equilibrium problems with identical stiffness matrix. To simplify the implementations and notations, the equations are rewritten in vector form by introducing the following parameters:

- The stiffness matrix:

$$\boldsymbol{K} = \sum_{e=1}^{m} w_e C_e B_{ej} B_{ei} = \boldsymbol{B}^T \boldsymbol{W} \boldsymbol{C} \boldsymbol{B} \tag{2.8}$$

- The displacement constraint vector:

$$\boldsymbol{U}(\boldsymbol{\epsilon^*}) = \sum_{e=1}^{m} w_e C_e \epsilon_e^* B_{ei} = \boldsymbol{B}^T \boldsymbol{W} \boldsymbol{C} \cdot \boldsymbol{\epsilon^*} \tag{2.9}$$

- The Lagragian multiplier constraint vector:

$$\boldsymbol{H}(\boldsymbol{\sigma^*}) = \sum_{e=1}^{m} w_e B_{ei} \sigma_e^* = \boldsymbol{B}^T \boldsymbol{W} \cdot \boldsymbol{\sigma^*} \tag{2.10}$$

where $diag(\boldsymbol{W}) = [w_1, .., w_m]^T$ and $diag(\boldsymbol{C}) = [C_1, .., C_m]^T$.

Using those formulations, the linear system in Equation 2.6 and 2.7 can be rewritten as:

$$\boldsymbol{K} \cdot \boldsymbol{u} = \boldsymbol{U}(\boldsymbol{\epsilon^*}) \quad \text{and} \quad \boldsymbol{K} \cdot \boldsymbol{\eta} = \boldsymbol{f} - \boldsymbol{H}(\boldsymbol{\sigma^*}) \tag{2.11}$$

The process of determining the optimal local data points $(\epsilon_e^*, \sigma_e^*)$ in the local datasets E_e is done iteratively. To begin, all bars in the truss are assigned random points $(\epsilon_e^{*(0)}, \sigma_e^{*(0)})$ from the corresponding local datasets E_e. The displacements $u_i^{(0)}$ and Lagrange multipliers $\eta_i^{(0)}$ are then calculated by solving the system of equations, and the stresses $\sigma_e^{(0)}$ are evaluated from the equations. For each member in the truss, the data points $(\epsilon_e^{*(1)}, \sigma_e^{*(1)})$ in E_e that are optimal with respect to the local state $(\epsilon_e^{(0)}, \sigma_e^{(0)})$ are identified. The iterations then continues by recursion until the local data assignments no longer change. All of these operations are summarised in Algorithm 1.

Algorithm 1 Data-driven solver

Data: Local datasets $\boldsymbol{E} = [E_1, ..E_m]$, $\boldsymbol{B}$-matrix. Applied loads $\boldsymbol{f}$

Initial local data assignment:
Set $k \leftarrow 0$
for *all* $e = 1, ..., m$ **do**
  Choose $(\epsilon_e^{*(0)}, \sigma_e^{*(0)})$ randomly from E_e
end

# Begin the DDCM loop
while ***True*** **do**

i) Solve for $\boldsymbol{u}^{(k)}$ and $\boldsymbol{\eta}^{(k)}$:

$$\boldsymbol{K} \cdot \boldsymbol{u}^{(k)} = \boldsymbol{U}(\boldsymbol{\epsilon}^{*(k)}) \quad \text{and} \quad \boldsymbol{K} \cdot \boldsymbol{\eta}^{(k)} = \boldsymbol{f} - \boldsymbol{H}(\boldsymbol{\sigma}^{*(k)})$$

ii) Compute local states:

$$\boldsymbol{\epsilon}^{(k)} = \boldsymbol{B} \cdot \boldsymbol{u}^{(k)} \quad \text{and} \quad \boldsymbol{\sigma}^{(k)} = \boldsymbol{\sigma}^{*(k)} + \boldsymbol{CB} \cdot \boldsymbol{\eta}^{(k)}$$

iii) Assign local state:
for *all* $e = 1, ..., m$ **do**
  $(\epsilon_e^{*(k+1)}, \sigma_e^{*(k+1)}) = \underset{(\epsilon'_e, \sigma'_e) \in E_e}{\operatorname{argmin}} \boldsymbol{W_e}(\epsilon_e^{(k)} - \epsilon'_e) + \boldsymbol{W_e^*}(\sigma_e^{(k)} - \sigma'_e)$
end

iv) Test for convergence:
if $(\boldsymbol{\epsilon}^{*(k+1)}, \boldsymbol{\sigma}^{*(k+1)}) = (\boldsymbol{\epsilon}^{*(k)}, \boldsymbol{\sigma}^{*(k)})$ **then**
  $\boldsymbol{u} = \boldsymbol{u}^{(k)}$
  $(\boldsymbol{\epsilon}, \boldsymbol{\sigma}) = (\boldsymbol{\epsilon}^{*(k)}, \boldsymbol{\sigma}^{*(k)})$
  exit while loop
else
  $k \leftarrow k + 1$
end
end

2.2 Numerical Analysis of convergence

The convergence of data-driven solvers with respect to the dataset is an important aspect to consider. In particular, when the materials in the truss follows a well-defined constitutive law, such as the Hook's law (Equation 2.1), it is expected that the data-driven solutions will converge to the classical solution as the datasets increasingly approximate the stress-strain curve.

To demonstrate this convergence property, different configuration of datasets are tested on the simple truss configuration, whose static system presented in Figure 2.2, which is made of a two nodded bar under a uniaxial load applied at node 2, with the displacement at node 1 being blocked. This type of configuration is commonly encountered in structural mechanics and can be easily solved using classical Finite Element Method (FEM) techniques.

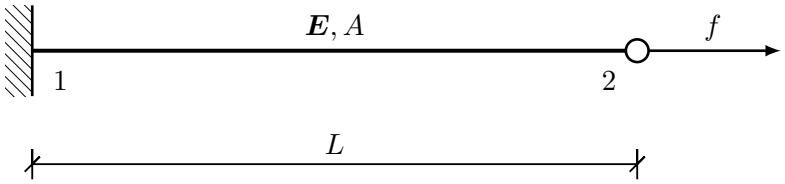

Figure 2.2: One dimensional bar under uniaxial load

For this system, the well-known equation for FEM, $\boldsymbol{K}_{FEM} \cdot [u_1, u_2]^T = [0, F]^T$, can be reduced to $\frac{EA}{L} u_2 = F$, where A is the cross-sectional area, E is the Young's modulus, L is the length of the bar, and f is the applied load. With $A, E, L, F = 1$, the displacement vector $\boldsymbol{u} = [0, 1]^T$ is obtained.

However, in the context of Data-Driven Mechanics, the material properties are not finite values, but rather a set of (ϵ, σ) pairs obtained from experiments or simulations. Since the task is to find the optimal pair that satisfies, or is closest to satisfying, the compatibility and equilibrium constraints, the datasets properties then become crucial for achieving meaningful results. In this project, the properties whose influence on the solver's convergence are evaluated are the dataset size and the noisiness.

- **Influence of the number of data points**

The number of data points in the set is an important factor because a large number of data points increases the chances of finding the optimal state.

To evaluate the effect of the number of data points on the accuracy of the data-driven solver, the relative error between the displacements computed with DDCM and FEM is calculated for various numbers of data points. In this study, the datasets assigned to the bar are collections of evenly spaced data points along the Hook curve, with the Young Modulus set to $E = 1$ [MPa], thus ensuring the results are not influenced by any noise.

The results, presented in Figure 2.3 provide insight into the relationship between the number of data points and the accuracy of the data-driven solver and can be used to guide the selection of the appropriate number of data points for a given application. As it could have been expected, the error decreases with the number of data points following a trend with a log slope of -0.999.It can be noted that this slope is similar to the results of Kirchdoerfer and Ortiz [1], which validates the result.

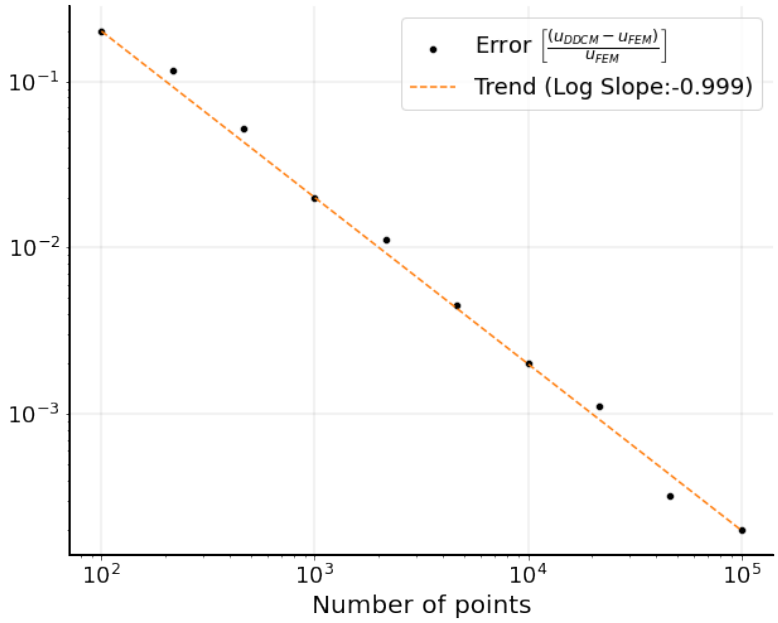

Figure 2.3: Effects of dataset size on the error on computed displacement

- **Influence of noisiness**

Noise is another important factor that influences the convergence of the DDCM solver. Here, it refers to the random dispersion in the dataset that deviates from the usual material law model. As a result, the computation can stray far from the traditional result, leading to a poor approximation of the true solution.

Figure 2.4 illustrates the influence of noise on the dataset, and how the points deviates from the expected material law model thus being a source of errors.

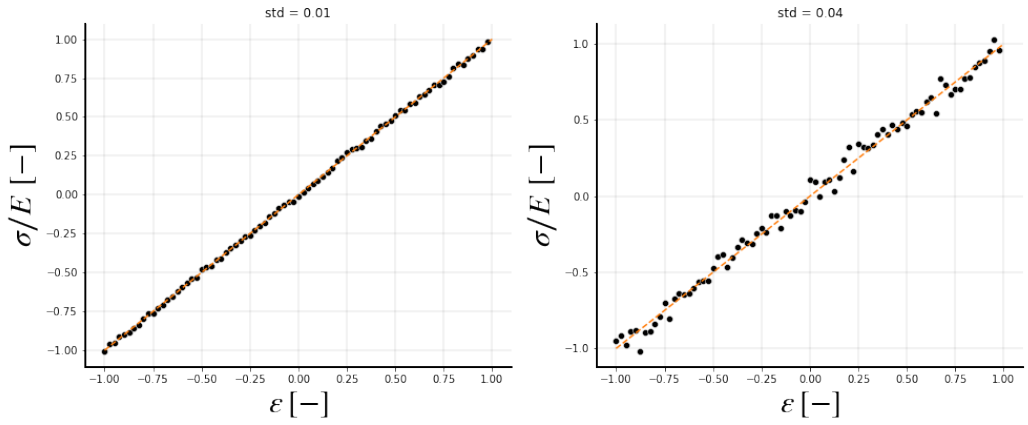

Figure 2.4: Effects of gaussian noise on the data points dispersion

To quantify the effect of noise on the DDCM solver, a study is conducted using a dataset with 1000 points and varying levels of noise, specifically in terms of standard deviation of a zero-mean gaussian distribution. The results, presented in Figure 2.5, demonstrate that as the noise

level increases, the error in the computed displacements also increases. Furthermore, it is observed that there appears to be a threshold around std=$1e^{-3}$ beyond which the error starts increasing noticeably. These results highlight the importance of controlling noise in the dataset to ensure accurate results when using a DDCM solver.

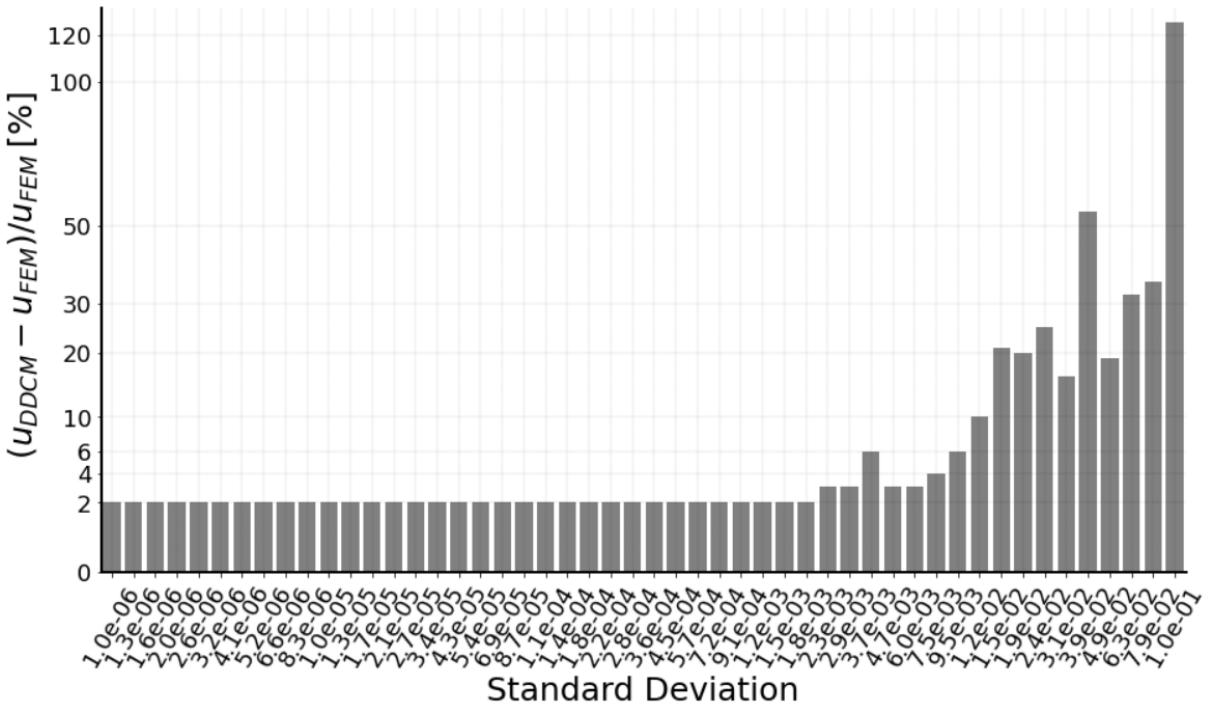

Figure 2.5: Effects of dataset noisiness on the error on computed displacement

- **Summary**

To summarise, it is crucial to consider the dataset noise and size factors when implementing a DDCM solver. Ideally, the dataset should be as large as possible to increase the chances of finding the optimal data point and as noise-free as possible to minimise the deviation from the usual material law model. This way, the DDCM solver can converge to the true solution and provide accurate results.

3 . Dynamic Data-Driven Solver

3.1 Adaptation of Static Solver

The extension of data-driven computing, presented in the previous chapter, to dynamical problems is demonstrated using the same example of truss structures as proposed by Kirchdoerfer and Ortiz in their second paper [2]. Essentially by using the same parameters that are introduced in the static scheme, the dynamic problem can be described by introducing the acceleration within the main equations.

In the traditional method the main system is the well-known harmonic equation:

$$\boldsymbol{M} \cdot \boldsymbol{a} + \boldsymbol{K} \cdot \boldsymbol{u} = \boldsymbol{F} \tag{3.1}$$

where $\boldsymbol{M}$ is the mass matrix, $\boldsymbol{K}$ the stiffness matrix and $\boldsymbol{a} = \dot{\boldsymbol{v}} = \ddot{\boldsymbol{u}}$ the acceleration.

This equation can be solved using finite element or other methods by introducing an adequate time discretisation to compute the acceleration $a_j = \ddot{u}_j$ at a given time t_j as a function of previous values of displacements. In this project, the chosen discretisation is based on the Newmark algorithm which allows to introduce the following Newmark predictors:

$$\boldsymbol{u_j^{pred}} = \boldsymbol{u_{j-1}} + \Delta t \cdot \boldsymbol{v_{j-1}} + \left(\frac{1}{2} - \beta\right) \Delta t^2 \cdot \boldsymbol{a_{j-1}} \tag{3.2}$$

$$\boldsymbol{v_j^{pred}} = \boldsymbol{v_{j-1}} + (1-\gamma)\Delta t \cdot \boldsymbol{a_{j-1}} \tag{3.3}$$

with the Newmark parameters β, γ and Δt being the time step.

These predictors allows to introduce the associated update for acceleration and velocity:

$$\boldsymbol{a_j} = \frac{1}{\beta \Delta t^2} \cdot (\boldsymbol{u_j} - \boldsymbol{u_j^{pred}}) \tag{3.4}$$

$$\boldsymbol{v_j} = \boldsymbol{v_j^{pred}} + \gamma \Delta t \boldsymbol{a_j} \tag{3.5}$$

Assuming the state $(\boldsymbol{\epsilon_j}, \boldsymbol{\sigma_j})$ of the truss at time t_j is known, the equilibrium constraint of the minimisation problem in Equation 2.5 becomes:

$$\sum_{e=1}^{m} w_e \cdot B_{ei} \cdot \sigma_{e,j} = f_{i,j} - \boldsymbol{M} \cdot a_{i,j} \quad \forall i \in \{1, ..., n\} \tag{3.6}$$

By inserting the expression in Equation 3.4 into 3.6, it results that in dynamic case the two independent equations in Equation 2.11 becomes coupled, as highlighted in the paper [2]. At time t_j the system becomes:

$$\begin{bmatrix} \boldsymbol{K} & -\frac{1}{\beta \Delta t^2}\boldsymbol{M} \\ \frac{1}{\beta \Delta t^2}\boldsymbol{M} & \boldsymbol{K} \end{bmatrix} \cdot \begin{bmatrix} \boldsymbol{u_j} \\ \boldsymbol{\eta_j} \end{bmatrix} = \begin{bmatrix} \boldsymbol{U}(\boldsymbol{\epsilon_j^*}) \\ \boldsymbol{f_j} - \boldsymbol{H}(\boldsymbol{\sigma_j^*}) + \frac{1}{\beta \Delta t^2}\boldsymbol{M} \cdot \boldsymbol{u_j^{pred}} \end{bmatrix} \tag{3.7}$$

Overall to solve the dynamic problem, the implemented algorithm is the same as for the static case by iterating on the number of time steps t replacing

3.2 Comparison with traditional solvers

First case: Same setup as in in Figure 2.2 in dynamics

To verify the accuracy and performance of the data-driven dynamic solver, it is tested and compared against a traditional dynamic finite element solver (FEM). To ensure consistency in the analysis, the same time discretisation is used for both solvers. Furthermore, the solvers are tested under a variety of load and truss configurations to provide a comprehensive evaluation.

The first studied case consists in testing the DDCM solver using the same static system depicted in Figure 2.2, with no initial displacement or velocity. The governing equation for this system is that of an harmonic oscillator, meaning the displacement should oscillate with a constant amplitude around the static solution.

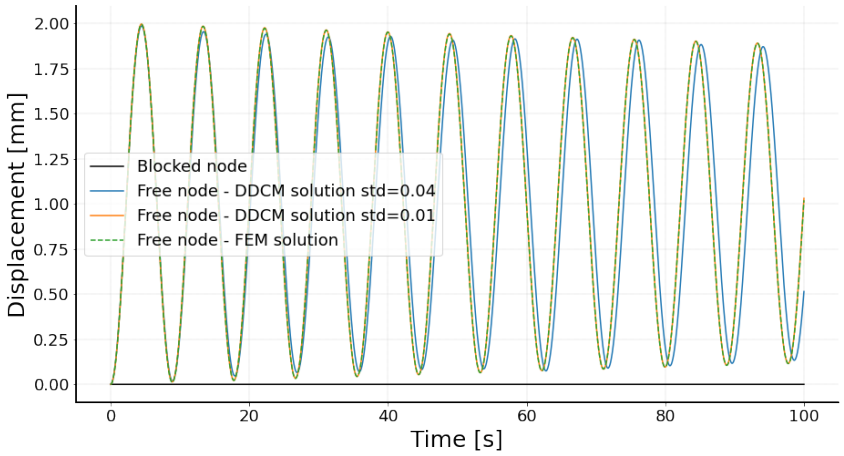

Figure 3.1: Displacements for the dynamic problem with the static system in Figure 2.2

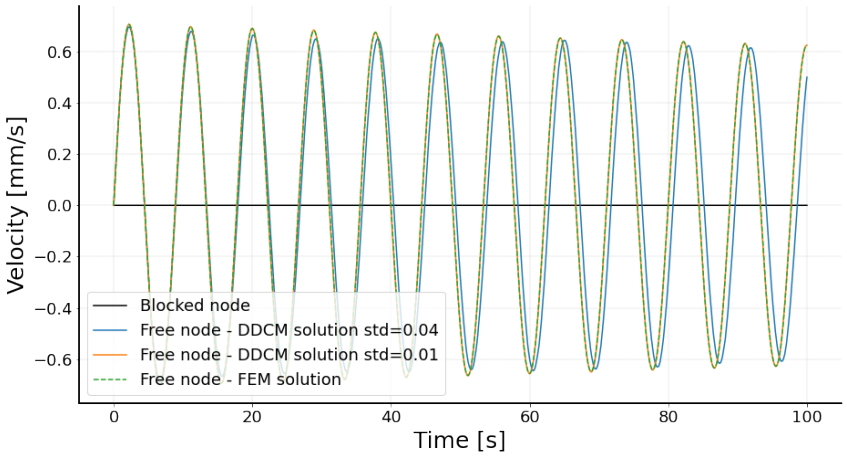

Figure 3.2: Velocities for the dynamic problem with the static system in Figure 2.2

The results of this comparison can be seen in Figure 3.1, where it can be observed that both the DDCM and FEM solutions follow the expected pattern of an harmonic oscillator. The velocities

are plotted in Figure 3.2 which confirms the result since the velocities oscillates at the same frequency as the displacements.

While the FEM solver and DDCM solver with lower noise (std=0.01) give similar results, the DDCM solution with a higher level of noise (std=0.04) is not in phase with the the other two. This behaviour can be attributed to the fact that even the static DDCM solver exhibits noticeable errors with such level of noise in the dataset. To furthermore highlight the influence of noise in the convergence to the classical solution, the displacements and velocities relative errors (compared to the FEM results) are shown in Figure 3.3 and 3.4.

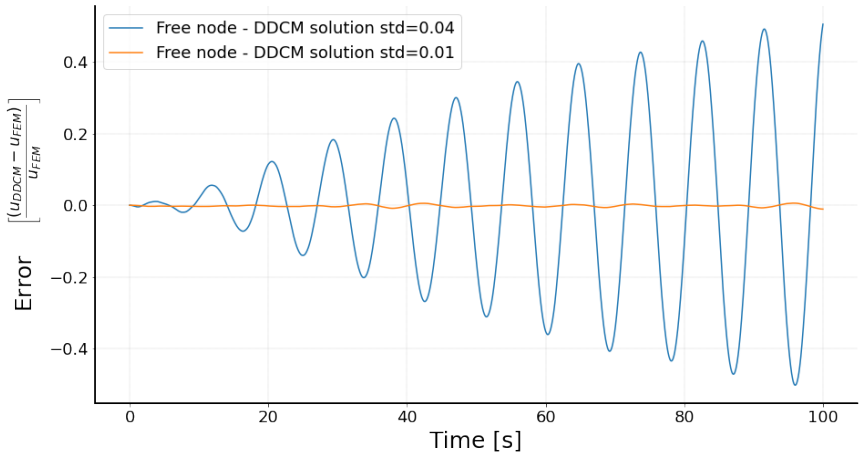

Figure 3.3: Displacements errors for the dynamic problem with the static system in Figure 2.2

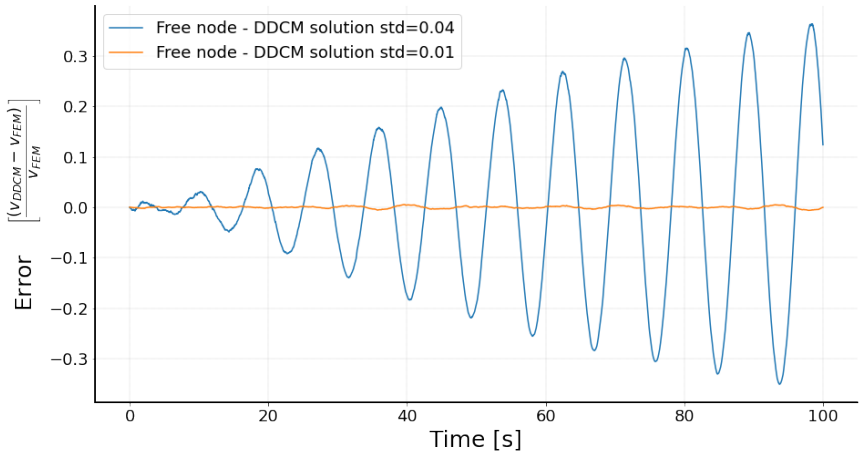

Figure 3.4: Velocities errors for the dynamic problem with the static system in Figure 2.2

Second case: With a linearly increasing force and three nodes

The second case evaluated uses the same static system as depicted in Figure 2.2 but with a free middle node and the applied force increases linearly over time, reaching a final value of $1 : [kN]$ at time $T = 400\ [s]$.

The expected displacement behaviour is a linear increasing curve. This is actually consistent with the results obtained from the FEM solver as shown in Figure 3.5. The data-driven solver, however, still exhibits some oscillations around the traditional results due to the noise added to the dataset. In Figure 3.6 the difference with the FEM results are presents, it can be seen that the error at the middle is slightly less than the one at the last node which can be explained by the fact the displacement at it is higher. As previously demonstrated, the closer the dataset is to the traditional material law, the more accurate the results of the DDCM will be towards the classical solution.

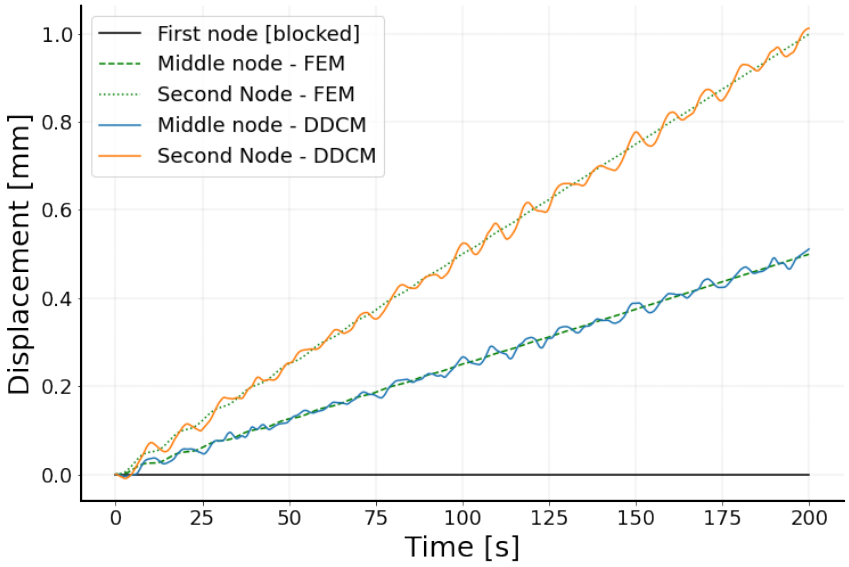

Figure 3.5: Displacement for the system which linearly increasing force, DDCM dataset with gaussian noise std=0.01

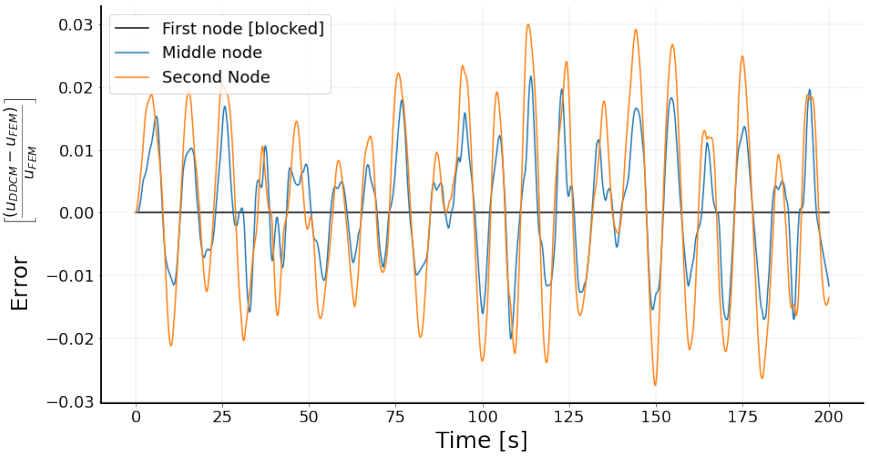

Figure 3.6: Displacement errors for the system which linearly increasing force, DDCM dataset with gaussian noise std=0.01

4 . Stick-Slip Modelling

4.1 Problem Description

The next phase of the project aims to study and solve the problem of a sliding block on a horizontal frictional surface, as shown in Figure 4.1.

Such behaviour is of interest in many engineering and physics applications. Friction plays a crucial role in this system, as it affects the force that is required to maintain the block in motion, as well as the force required to accelerate or decelerate the block.

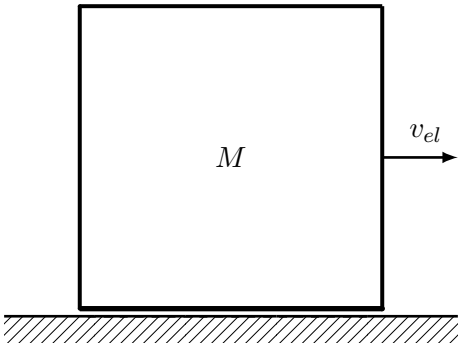

Figure 4.1: Sliding block schematisation

As the block moves, the frictional force fluctuates relatively to the block's velocity. At low speeds, the frictional force is relatively high, making it difficult to accelerate or decelerate the block. However, as the speed increases, the frictional force decreases towards a constant value, making it easier to accelerate or decelerate the block.

Here, in order to reuse the data-driven pipeline for trusses, the problem is modeled in one dimension using truss elements. So instead of a linear elastic truss bar acting as a stiff spring, the soil is represented by a bar that acts as a damping support and is linked to another bar representing the block, the corresponding system is shown in Figure 4.2

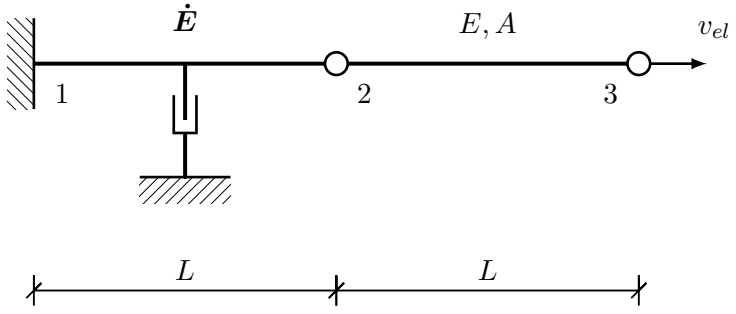

Figure 4.2: One dimensional modelisation of the sliding block (2-3) and frictional surface (1-2)

In this context, the phase space shifts from the traditional stress-strain pairs since the applied forces now relates to the velocity of the block. As such, the stress-strain relationship is replaced by the relation between stress σ and the variation of the strain with time, or the strain rate $\dot{\epsilon}$.

To describe the frictional sliding behaviour, a mathematical expression of the material law is then formulated as follows:

$$\sigma(\dot{\epsilon})/\sigma_0 = \begin{cases} \mu_s \cdot \dot{\epsilon}/\dot{\epsilon}_{sk} & \text{if } \dot{\epsilon} \leq \dot{\epsilon}_{sk} \\ \\ \mu_k + (\mu_s - \mu_k)\, e^{-\alpha \cdot (\dot{\epsilon} - \dot{\epsilon}_{sk})} & \text{otherwise} \end{cases} \tag{4.1}$$

where σ is the stress, μ_s and μ_k are the static and kinematic friction coefficients respectively, σ_0 is a reference stress, $\dot{\epsilon}$ is the strain rate, $\dot{\epsilon}_{sk}$ is a threshold strain rate at which the friction starts changing regime, and α is a coefficient that accounts for the rate at which the friction goes from μ_s to μ_k. The curve defined by such fuction is presented in Figure 4.3.

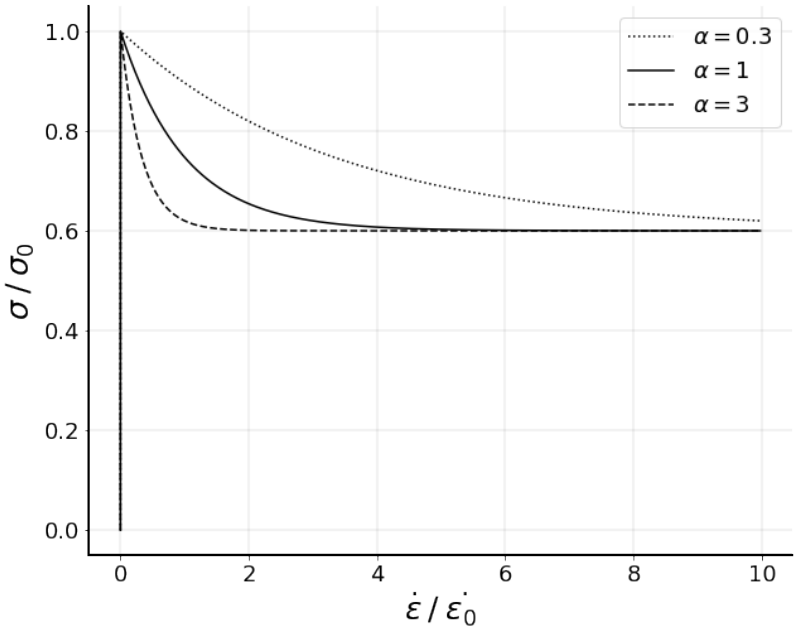

Figure 4.3: Representation of stress-strain rate relationship using Equation 4.1 with $\mu_s = 1$, $\mu_k = 0.6$ and $\dot{\epsilon}_{sk} \ll 1\ [s^{-1}]$

From the derived material law, it is possible to simulate the data-points required to test a data-driven solver. To develop it, the procedure for studying the system is divided into two steps, the first one being to develop the DDCM procedure that can solve problems in the phase space associated with the frictional surface and the second one being to adapt the solver to account for the sliding block.

4.2 Data-Driven Solver with damping

4.2.1 Theoretical formulations

In this section, the focus is directed towards a specific subsystem of the system presented in Figure 4.4. The objective is to create a data-driven solver that uses the relationship between stress and strain rate to calculate the displacement, velocity, and acceleration of the free node of the truss bar under uniaxial load.

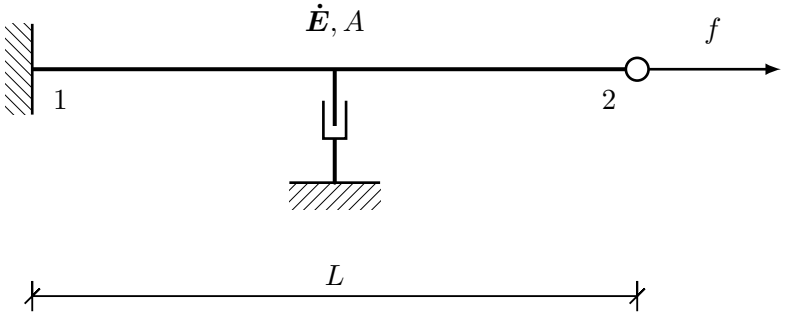

Figure 4.4: One dimensional bar with damping under uniaxial load

For these novel considerations, the constrained minimisation problem at time t_j becomes:

$$\begin{aligned} \text{Minimise:} \quad & \sum_{e=1}^{m} w_e \cdot \boldsymbol{F_{e,j}}(\dot{\epsilon}_{e,j}, \sigma_{e,j}) \\ \text{subject to:} \quad & \dot{\epsilon}_{e,j} = \boldsymbol{B_e v_j}, \quad \forall \,\text{bar}\, e \in \{1, ..., m\} \\ & \sum_{e=1}^{m} w_e \boldsymbol{B_e^T} \sigma_{e,j} = \boldsymbol{f_j} - \boldsymbol{M a_j} \end{aligned} \tag{4.2}$$

By using the expression of $\boldsymbol{v_j}$ and $\boldsymbol{a_j}$ derived in Equation 3.4 and 3.5, the system becomes only dependant on the nodal displacements. This allows us to enforce compatibility and equilibrium constraints, in the same manner as in Eq.5 of the paper [1], to highlight the stationary problem:

$$\delta \begin{bmatrix} \sum_{e=1}^{m} w_e \boldsymbol{F}_e \left(\boldsymbol{B_e v_j^{pred}} + \frac{\gamma \boldsymbol{B_e}(\boldsymbol{u_j} - \boldsymbol{u_j^{pred}})}{\beta \Delta t}, \sigma_{e,j} \right) \\ - \left(\sum_{e=1}^{m} w_e \boldsymbol{B_e^T} \sigma_{e,j} - \boldsymbol{f_j} + \frac{\boldsymbol{M} \cdot (\boldsymbol{u_j} - \boldsymbol{u_j^{pred}})}{\beta \Delta t^2} \right) \cdot \boldsymbol{\eta_j} \end{bmatrix} = 0 \tag{4.3}$$

Taking all possible variations, the following equations are obtained:

$$\delta \boldsymbol{u_j} \Rightarrow \boldsymbol{K u_j} - \frac{\boldsymbol{M \eta_j}}{\beta \Delta t^2} = \left(\sum_{e=1}^{m} w_e \boldsymbol{B_e^T} C_e \frac{\beta \Delta t}{\gamma} \dot{\epsilon}^*_{e,j} \right) - \boldsymbol{K} \left(\frac{\beta \Delta t}{\gamma} \boldsymbol{v_j^{pred}} - \boldsymbol{u_j^{pred}} \right), \tag{4.4}$$

$$\delta \boldsymbol{\eta_j} \Rightarrow \sum_{e=1}^{m} w_e \boldsymbol{B_e^T} \sigma_{e,j} = \boldsymbol{f_j} - \frac{\boldsymbol{M} \cdot (\boldsymbol{u_j} - \boldsymbol{u_j^{pred}})}{\beta \Delta t^2}, \tag{4.5}$$

$$\delta \sigma_{e,j} \Rightarrow C_e \boldsymbol{B_e \eta_j} + \sigma^*_{e,j}, \tag{4.6}$$

where $(\epsilon^*_{e,j}, \sigma^*_{e,j})$ denote the optimal data points for each of bar e and $\boldsymbol{K} = \left(\sum_{e=1}^{m} w_e \boldsymbol{B_e}^T C_e \boldsymbol{B_e} \right)$

By combining Equation 4.5 and 4.6 and rewriting the resulting equations in vector form, the system to be solved by the dynamic solver can be summarised as:

$$\begin{bmatrix} \boldsymbol{K} & -\frac{1}{\beta \Delta t^2} \boldsymbol{M} \\ \frac{1}{\beta \Delta t^2} \boldsymbol{M} & \boldsymbol{K} \end{bmatrix} \cdot \begin{bmatrix} \boldsymbol{u_j} \\ \boldsymbol{\eta_j} \end{bmatrix} = \begin{bmatrix} \boldsymbol{U} \left(\frac{\beta \Delta t}{\gamma} \boldsymbol{\dot{\epsilon}^*_j} \right) - \boldsymbol{K} \left(\frac{\beta \Delta t}{\gamma} \boldsymbol{v_j^{pred}} - \boldsymbol{u_j^{pred}} \right) \\ \boldsymbol{f_j} - \boldsymbol{H}(\boldsymbol{\sigma^*_j}) + \frac{1}{\beta \Delta t^2} \boldsymbol{M} \cdot \boldsymbol{u_j^{pred}} \end{bmatrix} \tag{4.7}$$

The solver can then be implemented similarly to the data-driven dynamic solver with some slight modifications, since Equation 4.7 is solved instead of Equation 3.7 and the new local states to be computed at time t_j and the k-th iteration of the DDCM loop are:

$$\boldsymbol{\dot{\epsilon}_j^{(k)}} = \boldsymbol{B} \cdot \boldsymbol{v_j^{(k)}} \quad \text{and} \quad \boldsymbol{\sigma_j^{(k)}} = \boldsymbol{\sigma_j^{*(k)}} + \boldsymbol{CB} \cdot \boldsymbol{\eta_j^{(k)}} \tag{4.8}$$

4.2.2 Numerical Results

In this study, the data-driven solver with damping is used to solve a friction problem in which the friction coefficient is constant and equal to the kinematic friction coefficient. To create the dataset, the mathematical expression in Equation 4.1 is used with a value of $\mu_s = \mu_k = 0.6$. With an initial velocity imposed, it is expected that the velocity would decrease to 0 and the displacement would increase until they reach a given plateau.

To provide a point of comparison, a traditional finite difference scheme is used to solve the damping problem of the form $\boldsymbol{Ma} + \boldsymbol{Cv} = \underline{0}$. The results of this traditional solver are then compared to the results of the data-driven approach, as presented in Figure 4.5 and 4.6.

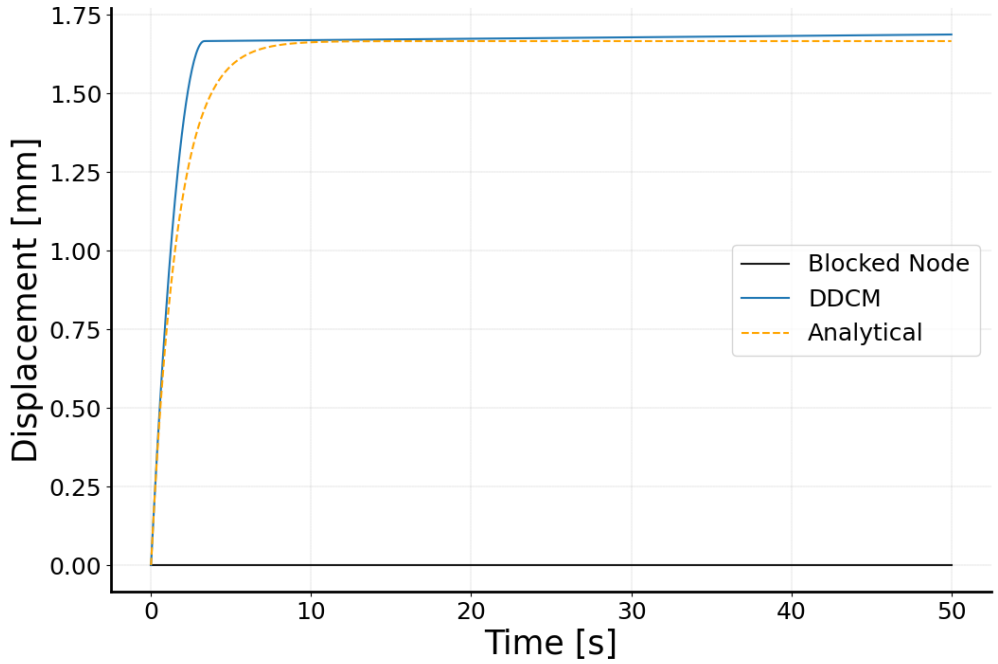

Figure 4.5: Displacements over time for the system of a bar with damping with an imposed initial velocity $v = 1\ [mm/s]$

The results of the computation using the data-driven solver with damping are consistent with the expected behaviour of the system, but there is a noticeable difference between the data-driven results and the traditional solver results. Despite the lack of noise in the dataset, the data-driven solver does not produce results that perfectly overlay the traditional solver results. Additionally, it can be observed that a large number of data points are needed to achieve reasonable accuracy, and using fewer points leads to a significant divergence in the results.

Despite these limitations, the data-driven solver is able to capture the general trend of the system, and therefore it can be used to model the stick-slip behaviour. Further validation and testing is necessary to confirm the validity of this data-driven solver with damping.

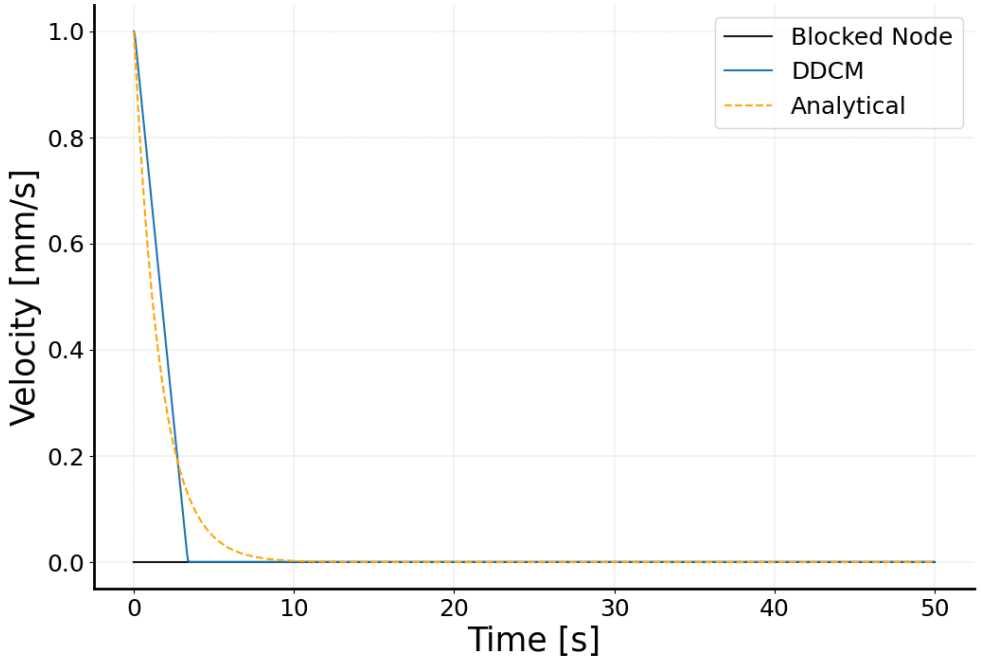

Figure 4.6: Velocities over time for the system of a bar with damping with an imposed initial velocity $v = 1\ [mm/s]$

4.3 Adaptation to Stick-Slip

4.3.1 Theoretical formulations

An effort is now made to extend the previously developed data-driven solver to include the dynamics of the sliding block, represented by bar (2-3) in Figure 4.2. To simplify the problem, it is assumed that the block has a constant imposed velocity, v_{el}, and as such the displacement of node 3 can be represented as $u_3 = v_{el} \cdot t$. With this assumption, the unknowns of the problem are limited to the nodal displacements, $\boldsymbol{u} = [0, u_2]^T$, of bar (1-2).

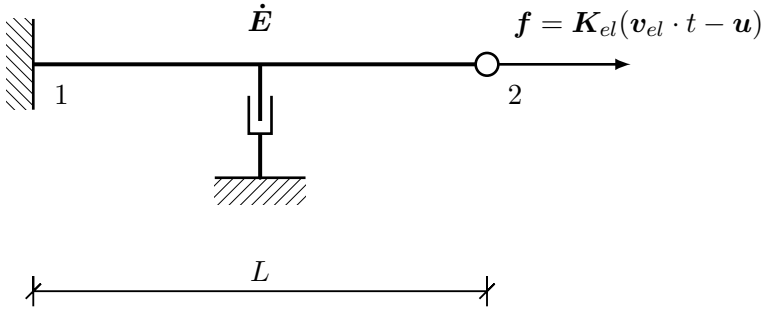

Figure 4.7: Modelisation of the sliding block as an equivalent force and the frictional surface as a truss bar with damping

To account for the influence of bar (2-3) of Figure 4.2 on the system represented in Figure 4.7, it is modeled as an equivalent spring force $\boldsymbol{f} = \boldsymbol{K}_{el}(\boldsymbol{v}_{el} \cdot t - \boldsymbol{u})$, where $(\boldsymbol{v}_{el} \cdot t - \boldsymbol{u})$ is the displacement difference between node 2 and 3, and $\boldsymbol{K}_{el}$ is the spring stiffness matrix.

This method allows for the incorporation of the dynamic behaviour of the sliding block while still utilising the previously developed data-driven solver for bar (1-2). The displacement, velocity, and acceleration of the truss bar's free node can be computed by solving the system of equations arising from the compatibility and equilibrium constraints, where the new expression of the force $\boldsymbol{f_j}$ is injected into the stationary problem Equation 4.2. Solving the problem with those new parameters leads to the final system at time t_j:

$$\begin{bmatrix} \boldsymbol{K} & -\left(\frac{\boldsymbol{M}}{\beta \Delta t^2} + \boldsymbol{K}_{el}\right) \\ \left(\frac{\boldsymbol{M}}{\beta \Delta t^2} + \boldsymbol{K}_{el}\right) & \boldsymbol{K} \end{bmatrix} \cdot \begin{bmatrix} \boldsymbol{u_j} \\ \boldsymbol{\eta_j} \end{bmatrix} = \begin{bmatrix} \boldsymbol{U}\left(\frac{\beta \Delta t}{\gamma} \dot{\boldsymbol{\epsilon}}_{\boldsymbol{j}}^{*}\right) - \boldsymbol{K}\left(\frac{\beta \Delta t}{\gamma} \boldsymbol{v_j^{pred}} - \boldsymbol{u_j^{pred}}\right) \\ \boldsymbol{K}_{el} \cdot \boldsymbol{v}_{el} t_j - \boldsymbol{H}(\boldsymbol{\sigma}_{\boldsymbol{j}}^{*}) + \frac{1}{\beta \Delta t^2} \boldsymbol{M} \cdot \boldsymbol{u_j^{pred}} \end{bmatrix} \tag{4.9}$$

4.3.2 Numerical Results

The final goal of the project is to use the solver developed for solving the system in Figure 4.7 to model the stick-slip behaviour between the block and the frictional surface. Stick-slip is a type of dynamic behaviour that is characterised by the block's displacement alternately sticking to and slipping on the surface. During the slipping phase, the block's displacement increases, while during the sticking phase, the displacement remains constant. As a result, here the displacement is expected to have a stair-like shape, with each step representing a slipping phase and each flat portion representing a sticking phase.

Additionally, the velocity is expected to oscillate around the imposed velocity value, with the amplitude of the oscillations being twice the imposed velocity value. This is due to the fact that during the sticking phase, the velocity is zero and during the slipping phase, the velocity reaches twice the imposed velocity value. As a result, the velocity curve is expected to have a sinusoidal shape, with the oscillations decreasing in amplitude as the block approaches the steady-state.

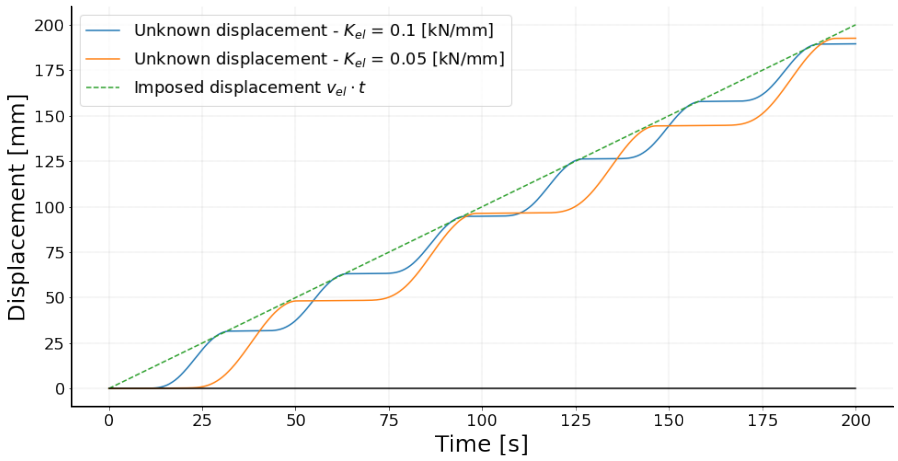

Figure 4.8: Displacements over time for the main system in Figure 4.7

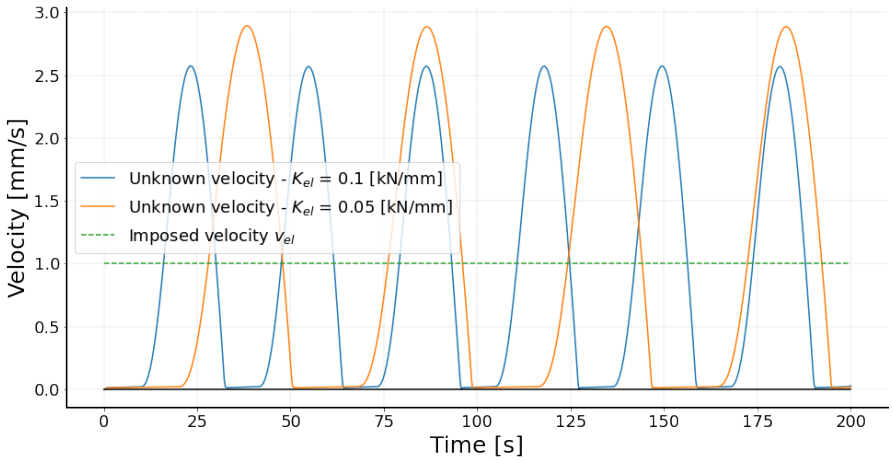

Figure 4.9: Velocities over time for the main system in Figure 4.7

The results presented in Figure 4.8 and 4.9 show that the solver is able to capture the stick-slip behaviour as the displacement and velocity follow the expected trend. The displacement curve is stair-like, with the block's displacement increasing during sticking phases and remaining constant during slipping phases and the velocity curve is sinusoidal as wanted. It can also be noted that a lower value of the spring stiffness $\boldsymbol{K}_{el}$ leads to longer sticking phases, as highlighted by the longer plateaus for both velocity and displacement on the figures and by the higher value of velocity. It can also be seen that the slipping phase is follows the same pattern since with a low stiffness the velocity reaches higher values and the displacement increasing portions lasts longer.

This indicates that the solver is able to accurately capture the stick-slip behaviour and can be utilised to model such systems. However, it is important to note that further verification are required to ensure the validity of the results, especially since the damping data-driven solver needs to be reassessed.

5 . Conclusion

In conclusion, the purpose of this project was to develop data-driven solvers for dynamic systems. A new solver was developed by utilising a relationship between stress and strain rate for frictional problems, which replaces the traditional stress-strain phase space.

To develop such solver, multiple steps were involved, starting from the basic data-driven solver which was tested and compared to a traditional finite element solver on a variety of load and truss configurations. The results showed that the data-driven solver was able to accurately capture the dynamic behaviour of simple trusses, with the accuracy increasing with the number of data points in the set and decreasing with the noise in the data.

By adapting this basic solver, it was possible to study the stick-slip behaviour between a block and a frictional surface, which resulted in a stair-like pattern of displacement and sinusoidal oscillations in velocity. The results obtained through the data-driven solver were found to be in good agreement with the expected behaviour, and thus it looks promising for modeling such systems in the future.

Bibliography

[1] T Kirchdoerfer and M Ortiz. 'Data-driven computational mechanics'. In: Comput. Methods Appl. Mech. Engrg. vol 304 (2016), pp. 81–101.

[2] T. Kirchdoerfer and M. Ortiz. 'Data-driven computing in dynamics'. In: International Journal for Numerical Methods in Engineering vol. 112.7 (2017), pp. 704–731. DOI: 10.1002/nme.5716.