Master's Thesis

Optimization Algorithms for Contact Mechanics

Sindhu Singh

Albert Ludwigs University of Freiburg

Faculty of Engineering

Department of Microsystems Engineering - IMTEK

Simulation Group

December 29th, 2020

Writing Period

29. 06. 2020 – 29. 12. 2020

Examiner

Prof. Dr. Lars Pastewka

Second Examiner

Prof. Dr. Moritz Diehl

Adviser

Antoine Sanner

Declaration

I hereby declare that I am the sole author and composer of my thesis and that no other sources or learning aids, other than those listed, have been used. Furthermore, I declare that I have acknowledged the work of others by providing detailed references of said work.
I hereby also declare that my Thesis has not been prepared for another examination or assignment, either wholly or excerpts thereof.

_____ _____

Place, Date Signature

Abstract

Numerical Optimization techniques are used to study the adhesive contact problem between an elastic solid and a rough surface. The optimization problems are formulated as constrained and unconstrained problems, using hard-wall or soft-wall constraints. This thesis examines the primal and the dual problem within the constrained problem scope. The performances of different algorithms are benchmarked against each other for low and high contact areas, using both the objectives. This thesis also introduces a preconditioner for unconstrained problem. This is solved using L-BFGS algorithm which performs better than all the other solvers when considered on an adhesive system with "instabilities".

Acknowledgements

Foremost, I would like to thank Prof. Dr. Lars Pastewka for providing me with the opportunity to work in the Simulation group at the institute for microsystems Engineering (IMTEK), Albert-Ludwigs University Freiburg. Thank you for the regular discussions during jour fixes and for reviewing my thesis work.

I also want to thank my second examiner Prof. Dr. Moritz Diehl for the co-examination of this thesis.

My sincere gratitude also to my supervisor Antoine Sanner, who always supported me and helped me during the whole thesis work. I am ineffably debted to him for devoting time to answer my questions and helping me stay focused on the work, during this time of pandemic. I learned a great deal working under him.

A big thank you to my family in India and my friends, for their continuous support during my Master's degree. A special thanks to Zeba Khan, Vanshaj Taxali, Daniel Noto, Rahul Pattnaik and many other friends who always helped me during my study work.

Contents

List of Figures

List of Tables

List of Algorithms

1 List of Acronyms

BEM Boundary Element Method 5

BUG Bugnicourt algorithm xiii

CG Conjugate Gradient 25

FIRE Fast Inertial Relaxation Engine 29

KKT Karush-Kuhn-Tucker conditions 17

PK Polonsky & Keer algorithm 26

L-BFGS Limited-memory BFGS algorithm 31

L-BFGS-B Limited-memory BFGS for bound constraints 31

MW-FIRE Mass Weighted Fast Inertial Relaxation Engine 2

1 Introduction

Understanding adhesion with rough surfaces is important for tires, seals, biomaterials, microcontact printing, and soft robotics. Both natural and man-made surfaces have roughness on small scales, and adhesion is highly influenced by this roughness [7] [8] [9]. The fundamental understanding of adhesion with rough surfaces, a grand scientific challenge, has significant technological applications. A human picking up an object and a gecko walking on the ceilings or running on a vertical wall rely on controlling adhesion. When measuring vital signs or delivering drugs, biomedical devices must adhere securely to the skin, but must then be removed without pain for disposal or reuse [10].

Contact mechanics simulations give insight on processes that are invisible for experiments, like unstable transitions between metastable states (instabilities). The large scale separation between interaction range, roughness and contact area requires efficient and scalable simulation algorithms. To date, the few in-depth comparisons between experimental results and accurate simulations have mainly focused on adhesionless contacts. Rigorous comparison are scarcer for adhesive interfaces. One such benchmark was made through the *contact mechanics challenge*, 2017 [11], where, the best performing algorithm by Bugnicourt et al. 2018 [2], took nearly three weeks on a single core to solve the simulation problem. This work tries to improve these simulation algorithms in order to make them faster.

In the continuum limit, the contact mechanics problem is a bound constrained

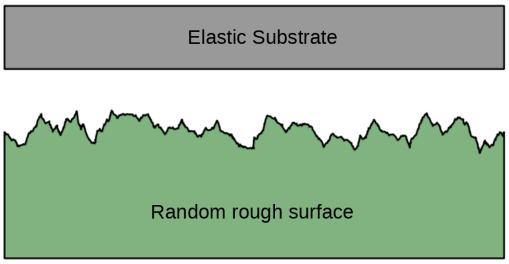


Figure 1: Elastic substrate interacting with a randomly rough surface.

problem. This constrained optimization problem is defined as primal and dual problems with the help of Karush-Kuhn-Tucker (KKT) conditions. By describing the system variables, *gap* and *pressure*, as the primal and the dual variable, respectively, a deeper understanding about the minimization behavior is achieved. These primal and dual problems are then solved using minimization algorithms developed by Bugnicourt et al. 2018 [2], Polonsky et al. 1999 [4] and the Limited-memory Broyden–Fletcher–Goldfarb–Shanno algorithm with Bound constraints (L-BFGS-B) from the Python package Scipy [12]. Their performances are benchmarked against each other for different adhesion and elasticity strengths.

The second approach is to model the system in the form of an unconstrained optimization problem by using a penalty function in the form of a repulsive interaction. This problem is solved using existing unconstrained optimizer: the Mass Weighted Fast Inertial Relaxation Engine (MW-FIRE) algorithm by Zhou et al. [3].

Taking motivation from the mass weighting, preconditioning by the elastic Green's function is introduced in L-BFGS algorithm to improve its performance

All these algorithms are also tested on instabilities, i.e., non-convex regions of the objective, to study the algorithms' robustness to handle them and determine their performances based on the function evaluations and walltimes.

In all the cases considered in this work, a new preconditioned L-BFGS algorithm is the best performing algorithm for solving the (unconstrained) rough contact problem

with adhesion.

Mathematical Notation

Within this thesis work, small bold letters are used to represent a vector quantity and small non-bold letter with index represents a particular value from that vector. For example: if $\mathbf{a}$ represents a vector then a_i is the i^{th} element of vector $\mathbf{a}$. Capital bold letters represent a matrix entity and capital non-bold letter represent an element of that matrix. For example: C_{ij} is the element of $\mathbf{C}$ at position i, j. Einstein summation is used to show the vector and matrix summations.

$\mathbf{x}[M]$ represents the values of $\mathbf{x}$ that are in the set M. $\sum \mathbf{x}[M]$ represents the sum of all the values in 'x' that are masked using values from M. $\mathbf{x}_k$ represents the value of $\mathbf{x}$ at k^{th} iteration. The iteration is represented by subscript 'k' only.

2 Theory

2.1 Formulation of the Physical Problem

This section elaborates on the mathematical description of the physical system.

2.1.1 Elasticity

Elasticity is the property of solid materials to return to their original shape and size after the forces deforming them have been removed. Every material in nature exhibits elasticity up to a certain extent.

For small displacements, stress depends linearly on displacements, allowing us to use a Green's function based approach, Boundary Element Method (BEM). BEM describes the elasticity of the surface of a halfspace. It reduces the dimensionality of the problem from three to two.

2.1.1.1 Green's Function

The Green's function $\mathbf{G}$ provides the important relationship in contact mechanics between the displacements ($\mathbf{u}$) and point forces ($\mathbf{f}$). For a one-dimensional system this can be described as,

$$u_i = G_{ij} f_j \tag{1}$$

and in the Fourier space ($\mathbf{Fu} = \tilde{\mathbf{u}}$), this can be written as, in Equation (2),

$$\widetilde{u}_{q_x} = \widetilde{G}_{q_x} \widetilde{f}_{q_x} \tag{2}$$

because G_{ij} is translationally invariant.

Same is also applicable for a two-dimensional system as well.

Even though the hessian product is defined as $\mathbf{Ku}$ (where, $\mathbf{K} = \mathbf{G^{-1}}$ is the stiffness matrix) in real space, in the actual implementations it is computed in Fourier space as $\mathbf{F^{-1}} \left(\widetilde{\mathbf{K}}\ \mathbf{F}\ \mathbf{u} \right)$, where F_{kl} represents a Fourier transform (i.e., $\exp(iq_k x_l)$) and F_{kl}^{-1}, the inverse of it ($= \exp(-iq_l x_k).\frac{1}{N}$), and where q_k represents the Fourier wave vector, $q_k = \dfrac{2\pi}{L_x} k$. L_x is the system size.

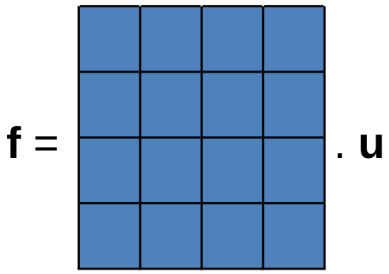
$$F_{ij} u_j = \tilde{u}_i \tag{3}$$

and

$$F_{ij}^{-1} \tilde{u}_j = u_i \tag{4}$$

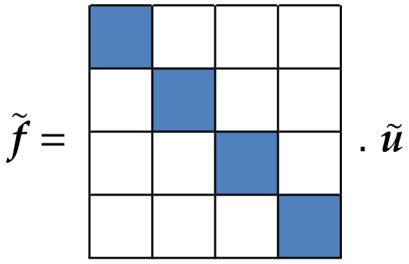
Figure 2 illustrates the dense nature of the real space Green's function matrix. Due to this, taking inverse of green's function matrix is a very costly operation. In order to avoid this, the Green's function matrix is formulated in Fourier space where, it is diagonal or block diagonal in nature. Same is illustrated in Figure 3.

In this work, the focus is only on normal contact i.e., only vertical displacements, and with Poisson's ratio $\nu = \frac{1}{2}$. This simplifies the Green's function matrix for periodic



(a) In real space

Figure 2: The Green's function matrix is dense in real space.



(b) In Fourier space

Figure 3: The Green's function matrix is diagonal in Fourier space.

systems as discussed by Campañá and Müser [13] and Pastewka et al. 2012 [14]. The relation between normal displacements and normal pressures can be written as,

$$\tilde{u}_{q_x^*} = \left(\frac{E^* q_x}{2}\right)^{-1} \tilde{p}_{q_x} \delta_{q_x - q_x^*} \tag{5}$$

At $q_x = 0$ the stiffness matrix becomes non invertible. To overcome this $\tilde{G}(q_0) = 2/q_1/E_s$.

It can be extended to non-periodic systems as described in the book on Contact Mechanics by Johnson K.L. [15] and further described using Fast Fourier Transform (FFT) by Pastewka and Robbins in [16]. Only periodic systems are considered for simplicity and because the precedent benchmarks were also mainly periodic.

From Equation (5), the elastic energy is then defined as Equation (6),

$$E_{el} = \frac{AE^*}{4} \sum_{qx} q_x |\tilde{u}_q x|^2 \tag{6}$$

where: E^* = effective modulus, $E^* = E/(1-\nu^2), E$ is the Young's modulus
A $= L^2$ where, L is the linear dimension of simulation cell
q $= 2\pi n/L$ where, n is number of grid points

For the ease of simulation, the system under study is modeled as a rigid rough indenter interacting with a flat elastic halfspace, which has a composite elastic moduli defined as,

$$E^* = \left(\frac{1-\nu_1^2}{E_1} + \frac{1-\nu_2^2}{E_2} \right)^{-1} \tag{7}$$

where: E_i = elastic moduli of the two materials
ν_i = Poisson ratio of the two materials

The elastic energy of the system is then defined as,

$$E_{el} = \frac{1}{2} u_i K_{ij} u_j \tag{8}$$

Figure 4 shows the simulation system with a flat rigid indenter. An important thing to note from this figure is, that it provides a relationship between *gap* $\boldsymbol{g}$ (array of gaps at each pixel value) and *displacement* $\boldsymbol{u}$ (array of displacements at each pixel value) i.e., $\mathbf{g} = \mathbf{u} - \mathbf{h}$, where $\mathbf{h}$ is an array of surface height at each pixel. Since $\mathbf{K}$ is positive definite by definition, the pressures are positive in the direction of displacement. And it can be noted from Figure 4, that when the motion is in the

half-space, the displacements are positive, meaning that the *repulsive pressures* are positive.

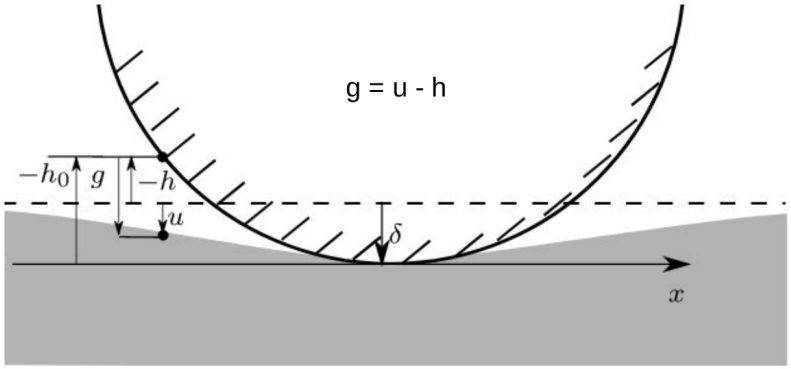
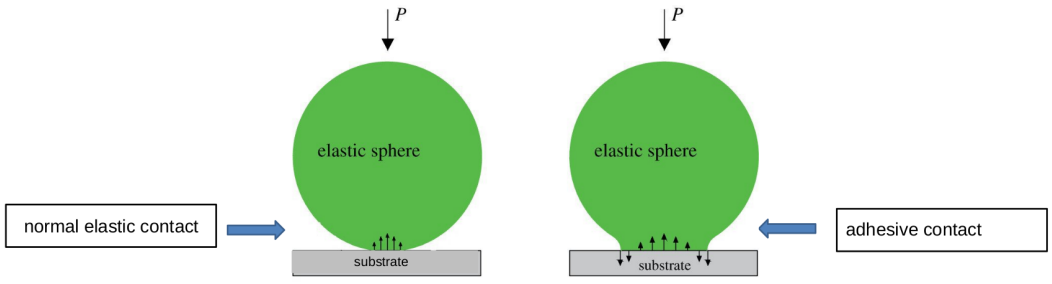


Figure 4: Spherical surface in contact with flat substrate depicting the directions for *gaps* g, *heights* h, *displacements* u and *penetration* δ .

2.1.2 Interaction: Adhesion

The continual fluctuations in the electronic distribution inside atoms or molecules trigger electromagnetic interactions. These interaction forces contribute to *Van der Waals forces* which are responsible for adhesion when the two surfaces are close to each other. Illustrated in Figure 5.



Source: https://royalsocietypublishing.org/doi/10.1098/rsif.2018.0738

Figure 5: Adhesive and non-adhesive (elastic) contact.

The energy of interaction between the two surface molecules coming in close vicinity with both adhesive and repulsive forces acting together can be described using the

Lennard-Jones potential as stated in Equation (9).

$$U = 4U_0 \left[\left(\frac{a}{r}\right)^{12} - \left(\frac{a}{r}\right)^{6} \right] \tag{9}$$

where: $U_0 = \dfrac{-C}{2r_0^6}$ at $r_0 = 1.12a$ and C is London's constant.

a = distance at which particle-particle potential is zero.

r = distance between the two interacting particles.

Using a volume integral gives the total energy of interaction between two bodies on a two dimensional surface in Equation (9). This computation can be made simpler by doing a surface integral. The potential is then described using 6-3 *Lennard-Jones potential* from Equation (10).

$$U = 4U_0 \left[\left(\frac{a}{r}\right)^{6} - \left(\frac{a}{r}\right)^{3} \right] \tag{10}$$

In continuum, the exact shape of the interaction potential do not matter. In this work, the interaction forces are expressed using the exponential model as described by Müser in 2014 [17] and originally by Fisher et al. 1969 [18].

Adhesive energy per pixel in simulation environments is then written as,

$$E_{exp} = -\gamma \exp(-g/\rho) \tag{11}$$

where: γ = coefficient of attraction or attractive energy per area.

ρ = interaction range.

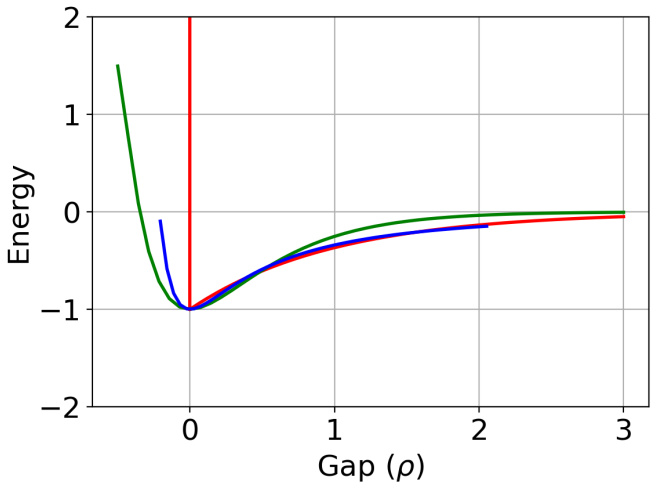


Figure 6: Comparison between different interaction potentials: (i) Lennard - Jones potential (Equation (10)) in blue, (ii) repulsive exponential (Equation (12)) in green, (iii) Exponential with hardwall repulsion (Equation (11)) in red

.

and the per pixel repulsive energy is defined as,

$$E_{exp} = \gamma \exp(-2g/\rho) \tag{12}$$

where: γ = coefficient of repulsion or repulsive energy per area.

ρ = interaction range.

The total adhesive energy of a one-dimensional system is the sum of the energies of all the pixels multiplied by the area per pixel and is defined as,

$$E_{int} = \frac{L}{N_x} \sum_n \gamma_1 \exp(-2g_n/\rho_1) - \gamma_2 \exp(-g_n/\rho_2) \tag{13}$$

where: N_x = grid size.

g_n = gap.

γ_1 = repulsive energy per area.

γ_2 = attractive energy per area.

ρ_1 = interaction range for repulsion.

ρ_2 = interaction range for attraction.

In general the interaction potential of the whole surface can be written as, $\sum_i \phi(g_i)$. So, differentiating it once with respect to gap gives the interaction force i.e., $\phi'(g_i)$ and differentiating twice with respect to gap, gives the Hessian as $\phi''(g_i)\delta_{ij}$.

The adhesive strength of the system can be represented using *Tabor coefficient*, μ. Mathematically, $\mu \propto \lambda$ (Maugis parameter).

$$\lambda = \frac{2\sigma_0}{(\pi\omega E^{*2}/R)^{1/3}} \tag{14}$$

where, R is the radius of the sphere or of asperities in the rough contact.

In Equation (14) the numerator represents the maximum adhesive force and the denominator is the typical repulsive pressure for a non-adhesive contact. For a system with adhesive interaction, the Hessian is $K_{ij} - \phi''(g_i)\delta_{ij}$. If in the Hessian, stiffness matrix (K_{ij}) remains the dominant part then, the system is considered non-adhesive or weakly adhesive and if, the adhesion part ($\phi''(g_i)\delta_{ij}$) dominates then, the system is called strongly adhesive. So the systems with high Tabor parameter are strongly adhesive.

Strongly adhesive interaction can lead to negative hessian values which means there exists several minima. This leads to *instabilities*, and causes hysteresis in the loading unloading curve.

2.2 Formulation of the Optimization Problem

Our general contact mechanics problem can be formulated in both ways, either as an *unconstrained optimization problem* or as a *constrained optimization problem.* The next two upcoming sections will define these optimization problems.

2.2.1 Unconstrained Optimization Problem

The unconstrained problem consists of the minimization of the total energy. The total energy consists of the elastic energy, computed in Fourier space and the sum of interaction energies over all the pixels, where the interaction energy includes a repulsive component as in Equation (13),

$$\min_{\mathbf{u}} E_{tot} = \frac{1}{2}(F_{ij}u_j)^* \widetilde{K}_{il}(F_{lm}u_m) + \sum_i \phi(u_i - h_i) \tag{15}$$

For $\rho_1 \to 0$, Equation (13) is equivalent to hard-wall repulsion.

2.2.2 Constrained Optimization Problem

2.2.2.1 Non-Adhesive Problem

The contact problem is described as hard-wall constraint problem with $\mathbf{g} > 0$, which means that there is no interpenetration between the contacting surfaces. At the static equilibrium only repulsive forces exist, i.e., $\mathbf{p} > 0$. The complementarity is given by $\mathbf{p^T g} = 0$, showing that there are only repulsive interactions when the contact surfaces touch .

The contact problem considering only the elastic forces (as depicted in Figure 4) is defined as stated in Equation (16),

$$\begin{aligned} \min_{\mathbf{u}} E_{el} =& \frac{1}{2}\mathbf{u^T K u} \\ \text{subject to,} & \\ & u_i \geq h_i \end{aligned} \tag{16}$$

where: $\mathbf{u}$ = displacements
$\mathbf{K}$ = stiffness matrix
$\mathbf{h}$ = surface heights

Since it is a constrained programming problem, it can be defined in two forms: as a primal problem or as a dual problem. In the next sections, these formulations are shown and discussed.

2.2.2.2 Primal Problem

Primal problem considered in further calculations is computed using Equation (16). To simplify the constraints, it is expressed in terms of gaps ($\mathbf{g} = \mathbf{u} - \mathbf{h}$) in Equation (17) ,

$$\begin{aligned} \min_{\mathbf{g}} E_{el} &= \frac{1}{2}(\mathbf{g}+\mathbf{h})^T\mathbf{K}(\mathbf{g}+\mathbf{h}) \\ &= \frac{1}{2}\mathbf{g}^T\mathbf{K}\mathbf{g} + \mathbf{g}^T\mathbf{K}\mathbf{h} + \frac{1}{2}\mathbf{h}^T\mathbf{K}\mathbf{h} \\ \text{subject to,} & \\ g_i &\geq 0 \end{aligned} \tag{17}$$

2.2.2.3 Dual Problem

To formulate the dual problem, first, Lagrangian is taken of Equation (17) to include the inequality constraints inside the equation and then, the $infimum$ with respect to gap is taken ($-q(\mathbf{p}) = \inf_g \mathcal{L}(\mathbf{g}, \mathbf{p})$) and is then multiplied with a minus to make it again a *minimization problem* which gives the resulting dual objective in Equation (21).

$$\mathcal{L}(\mathbf{g}, \mathbf{p}) = \frac{1}{2}\mathbf{g}^T\mathbf{K}\mathbf{g} + \mathbf{g}^T\mathbf{K}\mathbf{h} + \frac{1}{2}\mathbf{h}^T\mathbf{K}\mathbf{h} - \mathbf{p}^T\mathbf{g} \tag{18}$$

which gives,

$$\partial_{\mathbf{g}}\mathcal{L} = \mathbf{K}\mathbf{g} - \mathbf{p} + \mathbf{K}\mathbf{h} \tag{19}$$

$$\partial_{\mathbf{p}}\mathcal{L} = -\mathbf{g} \tag{20}$$

The corresponding dual objective for the primal objective defined in Equation (17) is defined in Equation (21),

$$\begin{aligned} &\min_{\mathbf{p}} q(\mathbf{p}) = \frac{1}{2}\mathbf{p}^T\mathbf{K}^{-1}\mathbf{p} - \mathbf{p}^T\mathbf{h} \\ &\text{subject to,} \\ &\qquad p_i \geq 0 \end{aligned} \tag{21}$$

where: $\mathbf{p}$ is pressure.

2.2.2.4 Mean Value Constraint

An additional constraint is introduced to enforce the mean value on the vector of unknowns $\mathbf{g}$ or $\mathbf{p}$. Adding the mean pressure constraint (in a dual problem) is useful because it is what is often prescribed in experiments, also it helps overcome the additional problem of the mean value of $\mathbf{u}$ that is the stiffness (at $q = 0$) being ill-defined. Equation (16) is updated with additional constraint,

$$\begin{aligned} \min_{\mathbf{u}} E_{el} =& \frac{1}{2}\mathbf{u}^T\mathbf{K}\mathbf{u} \\ \text{subject to,} & \\ & u_i \geq h_i \quad \text{and} \\ & \frac{\sum \mathbf{u}[J]}{N_J} = z \end{aligned} \tag{22}$$

where, z is the mean value that should be enforced and N_J is the number of points in $\mathbf{u}$.

J is the set of all points. The equations are split into the contacting regions (set of points I) and non-contacting regions (set of points in set M), where $M = J \setminus I$. Set of points I represents the active set and set M represents the non-active set.

The next steps are done in order to understand and formulate the constraints inside the conjugate gradient algorithms used in this work.

Lagrangian of Equation (22) is taken to include the constraints inside the objective,

$$\mathcal{L}(\mathbf{u}, \lambda, \boldsymbol{\mu}) = \frac{1}{2}\mathbf{u}^T\mathbf{K}\mathbf{u} - \lambda\left(\frac{\sum \mathbf{u}[J]}{N_J} - z\right) - \boldsymbol{\mu}^T(\mathbf{u} - \mathbf{h}) \tag{23}$$

where, λ (a scalar value) and $\boldsymbol{\mu}$ (a vector) are *Lagrangian* variables.

For solving Equation (23), Karush-Kuhn-Tucker conditions (KKT) system is computed. KKT conditions are the necessary conditions that provide a minimum for the given constrained problem. In the following calculations, the same are computed for the non-adhesive system.

KKT conditions[1]: For a system defined as

$$\begin{aligned} &\min_{x} \quad f(x) \\ &\text{subject to} \quad h(x) \geq 0 \end{aligned} \tag{24}$$

the KKT conditions are (for $i = 1, \ldots, q$) :

1. $\nabla f(x) - \sum_{i=1}^{q} \nabla h_i(x)\mu_i = 0$

2. $h_i(x) \geq 0$

3. $\mu_i \geq 0$

4. $\mu_i h_i(x) = 0$

Equation (23) gives,

$$\frac{\partial \mathcal{L}}{\partial \lambda} = \frac{\sum \mathbf{u}[J]}{N_J} - z \tag{25}$$

$$\frac{\partial \mathcal{L}}{\partial \boldsymbol{\mu}} = \mathbf{u} - \mathbf{h} \tag{26}$$

$$\frac{\partial \mathcal{L}}{\partial \mathbf{u}} = \mathbf{K}\mathbf{u} - \frac{\lambda}{N_J} - \boldsymbol{\mu} = 0 \tag{27}$$

Solving Equation (27) for $\boldsymbol{\mu}[I]$ gives,

$$\begin{aligned} \boldsymbol{\mu}[I] &= -\frac{\lambda}{N_J} + (\mathbf{K}\mathbf{u})[I]) \\ \boldsymbol{\mu}[M] &= 0 \end{aligned} \tag{28}$$

To determine λ, the mean of Equation (27) is taken,

$$\begin{aligned} 0 &= \sum \left((\mathbf{K}\mathbf{u})[J] - \frac{\lambda}{N_J} - \boldsymbol{\mu}[J] \right) \\ 0 &= \sum (K\mathbf{u})[M] + \sum (K\mathbf{u})[I] - \sum ((K\mathbf{u})[I] - \frac{\lambda}{N_J}) - \lambda \\ \lambda &= \sum (K\mathbf{u})[M] + \frac{N_I \lambda}{N_J} \\ \frac{N_M}{N_J}\lambda &= \sum (K\mathbf{u})[M] \\ \lambda &= \frac{N_J}{N_M} \sum (K\mathbf{u})[M] \end{aligned} \tag{29}$$

where: N_J = is the number of points in J.

N_M = is the number of points in M.

then,

$$\frac{\partial \mathcal{L}}{\partial \mathbf{u}[M]} = (K\mathbf{u})[M] - \frac{\sum (K\mathbf{u})[M]}{N_M} \tag{30}$$

and

$$\begin{aligned} \boldsymbol{\mu}[I] &= \frac{\sum (K\mathbf{u})[I]}{N_I} - (K\mathbf{u})[I] \\ &= 0 \qquad otherwise. \end{aligned} \tag{31}$$

The residual $\mathbf{r}(\mathbf{u}) = \dfrac{\partial \mathcal{L}}{\partial \mathbf{u}}(\mathbf{u}, \lambda(\mathbf{u}), \mu(\mathbf{u}))$. The minimizer works upon finding $\mathbf{u}$ through the iteration process. Factors $\lambda(\mathbf{u})$ and $\boldsymbol{\mu}(\mathbf{u})$ are computed analytically in Equation (29) and Equation (31).

If Equation (29) is written as,

$$\frac{\lambda}{N_J} = \frac{\sum (\mathbf{K}\mathbf{u})[M]}{N_M} \tag{32}$$

we can say that the mean of the points that are out of contact is to be considered.

To make the notation more generic, $\mathbf{H}$ is used to represent the Hessian ($\mathbf{K}$ or $\mathbf{K}^{-1}$) and variable $\mathbf{x}$ to represent the optimizing variable, instead of $\mathbf{p}$ or $\mathbf{g}$.

2.3 Numerical Optimization

After formulation of the optimization problem for the system in hand, the next step is to solve these problems.

To solve these optimization problems, optimization algorithms (or iterative solvers) are used to find the *minimum*. These optimization algorithms start with an initial guess and iterate. Using this initial guess, a sequence of iterates is generated until they satisfy the convergence condition of desired tolerance. Some use all the history of descent directions to estimate the next iterates (e.g.: conjugate directions (Algorithm (1) in appendix) with Gram-Schmidt approach), and in some solvers, just the history of last iterate is enough to make the best prediction for the next iteration (e.g.: conjugate gradients algorithm).

A good optimization algorithm is the one that can find the minimum with high accuracy, in the least amount of time, and that is robust against locally negative definite Hessians. To achieve this, different techniques are used to make the optimization problem less difficult to handle for the solvers.

For example:

Given a quadratic system,

$$\min_{\mathbf{x}} \phi(\mathbf{x}) = \frac{1}{2}\mathbf{x}^T\mathbf{A}\mathbf{x} - \mathbf{b}^T\mathbf{x} \tag{33}$$

the gradient is,

$$\mathbf{Ax} - \mathbf{b} \tag{34}$$

and the Hessian is **A** (a positive definite matrix).

When A is a diagonal matrix, then contours of the quadratic system are aligned with the coordinate directions, as shown in Figure 7. Then, by using a *Conjugate direction* Algorithm (1) from appendix, in maximum 'n' (length of x) iterations, the minimum for the system defined in Equation (33) can be found.

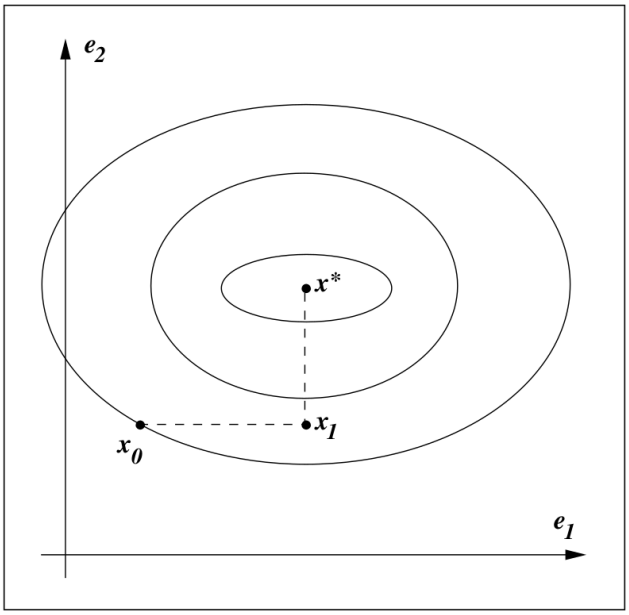


Figure 7: Successive minimization along coordinate axis will give a solution in n iterations with diagonal Hessian matrix. Reprinted from *Numerical Optimisation*(p.104), by J. Nocedal and S.J. Wright[1].

When the Hessian matrix A is not diagonal, the contour ellipses are not aligned with the coordinate axis so the solution minimum can not be achieved in n iterations by using *Conjugate direction* Algorithm (1) in the appendix. This is illustrated in Figure 8.

To solve this problem, a technique called preconditioning is used. Simplest preconditioning can be understood as the Hessian which is updated by using $L^{-1}A$ as the

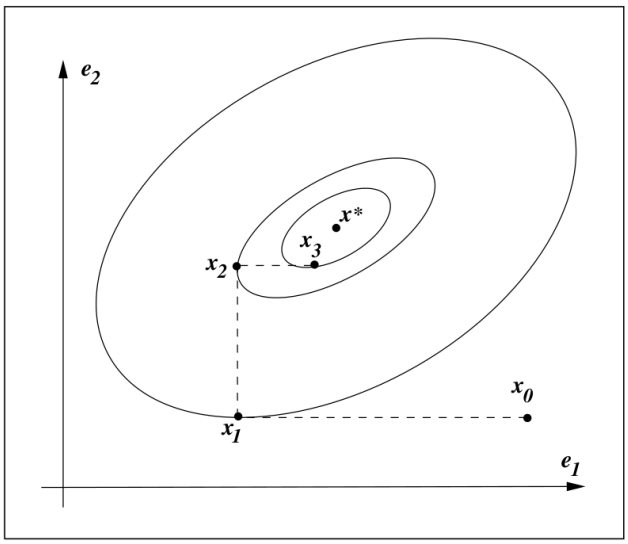


Figure 8: Successive minimization along coordinate axis will not give solution in n iterations. Reprinted from *Numerical Optimisation*(p.105), by J. Nocedal and S.J. Wright[1].

new Hessian, resulting in a sparse matrix with improved spectral properties. L is called the preconditioning matrix.

The process of finding a proper preconditioner can be tricky. More about this is discussed in the next Section 2.3.1 and how it helps to improve the problem in hand (Section 4.2.2).

2.3.1 Preconditioning

Preconditioning is a technique used to get better spectral properties for Hessian ($\mathbf{A}$), by multiplying a preconditioning matrix (L^{-1}) to the gradients.

The condition number describes the sensitivity of the solution to small perturbations in the input data. For example, for an non-singular matrix $\mathbf{A}$, it is given by $\kappa(A) = \lambda_{\max}/\lambda_{min}$. Where, λ_{max} and λ_{min} are the maximum and minimum eigen values of

A. Condition number determines the speed of convergence for an algorithm. The lower the condition number, the higher the speed of convergence.

In general, the condition number for a function f from $\mathbb{R}^n$ to $\mathbb{R}^n$ is given by,

$$cond(f, x) = \frac{\|x\|\|J(x)\|}{\|f(x)\|} \tag{35}$$

where: $J_{ij}(x) = (\partial f_i(x)/\partial x_j)$

As described in the book by Van der Voorst on *Iterative Krylov Methods* [19], the linear operator L has the following properties:

For a given (ill-conditioned) system,

$$Ax = b \tag{36}$$

1. L is a good approximation to A.

2. The cost of construction of L is not prohibitive.

3. The system $Ly = z$ is easier to solve as compared to the original system, $Ax = b$.

In a more formal way, it can be said that preconditioning is done so that the new condition number of the preconditioned matrix is lower than the condition number for the original matrix.

Ideally, the preconditioned system looks like :

$$L^{-1}Ax = Ix = L^{-1}b \tag{37}$$

There are three different ways in which a preconditioner can be applied that will lead to same eigenvalues. However, their convergence behavior may differ because of differences in their eigenvectors.

1. **Left-preconditioning**: The optimization algorithm is applied to Equation (38),

$$L^{-1}Ax = L^{-1}b \tag{38}$$

2. **Right-preconditioning**: Right-preconditioning affects only the Hessian and not the right hand side. So we get Equation (39),

$$AL^{-1}y = b \tag{39}$$

   where: $x = L^{-1}y$.

   Right preconditioning can also be interpreted as, substitution of x with $L^{-1}y$ and then solving for the updated variable y.

3. **Two-sided preconditioning**: If the preconditioner L can be defined as $L = L_1L_2$, then the preconditioned system can be defined as,

$$L_1^{-1}AL_2^{-1}z = L_1^{-1}b \tag{40}$$

   where: $x = L_2^{-1}z$.

Preconditioning is then applied to the contact mechanics problem in hand, in Fourier space. This is discussed later in the Results section.

3 Methods

3.1 Optimization Algorithms

There are many algorithms that have been developed over time to solve the rough contact problem with different approaches. Either, the contact mechanics problem is formulated as a minimization problem with multiple constraints and solving it using popular iterative methods like Constrained Conjugate Gradient (Conjugate Gradient (CG)) as discussed by Vollebregt [20], or using Limited memory Broyden-Fletcher-Goldfarb-Shanno Bounded (L-BFGS-B). Alternately, using the molecular dynamics approach where the contact problem is formulated using a physical analogy such as a simple damped harmonic oscillator or as a skier going downhill [5] [6]. The objective is to find an efficient iterative solver that can solve the contact problem with highest accuracy, while being computationally efficient and robust in handling strong adhesion and negative Hessians.

Few selected algorithms are compared here,

- Constrained Conjugate Gradient algorithm by Polonsky & Keer 1999 [4].
- Constrained Conjugate Gradient algorithm by Bugnicourt et al. 2018 [2].
- GFMD (Green's Function Molecular Dynamics) described in Prodanov et al. 2013 [5].

- FIRE (Fast Inertial Relaxation Engine) by Zhou et al. 2019 [3] and Bitzek et al. 2006 [6].

- Mass weighted FIRE by Zhou et al. 2019 [3].

- L-BFGS-B as implemented in Python package Scipy [12].

The detailed implemented versions of the algorithms are described as a pseudo-code in the appendix.

Considering a general description of the contact problem using $\mathbf{x}$ in place of $\mathbf{g}$ or $\mathbf{p}$, and a Hessian matrix ($\mathbf{H}$) in order to avoid switching between $\mathbf{K}$ and $\mathbf{K}^{-1}$,

$$E = \frac{1}{2} x_i H_{ij} x_j \tag{41}$$

3.1.1 Constrained Conjugate Gradient by Polonsky & Keer

As discussed in Section 2.2.2, there are two ways in which the contact mechanics problem can be formulated. The constrained conjugate gradient algorithm by Polonsky & Keer 1999 [4] (PK) tries to solve the dual problem with pressure as variable, as defined in Section 2.2.2.3.

In Equation (21), the problem that is solved by Polonsky & Keer algorithm (PK) is described. In addition to solving Equation (21), PK algorithm also handles the mean value constraint as described in Equation (22) (same is applicable here with pressure $\mathbf{p}$ as variable).

As compared to a standard conjugate gradient algorithm as discussed in appendix Algorithm (2), the parameters *step length*, $\alpha_k = \dfrac{\mathbf{r}_k^T \mathbf{r}_k}{\mathbf{d}_k^T \mathbf{H} \mathbf{d}_k}$, *coefficient for computing conjugate direction*, $\beta_{k+1} = \dfrac{\mathbf{r}_{k+1}^T \mathbf{r}_{k+1}}{\mathbf{r}_k^T \mathbf{r}_k}$ and *descent direction*, $\mathbf{d}_{k+1} = -\mathbf{r}_{k+1} + \beta_{k+1} \mathbf{d}_k$ are defined according to the Equation (42), (43) & (44).

The Lagrangian multipliers introduced in Equation (29) and Equation (31) are implemented by *masking* the sum. I_g is the set of all the points and I_c is the set of points in contact i.e., the active set for the primal problem. For the dual problem, $\neg I_c$ (logical NOT of I_c) is the active set.

$$\alpha_k = \frac{\mathbf{x}_k^T[I_c]\mathbf{d}_k[I_c]}{(\mathbf{Hd})_k^T[I_c]\mathbf{d}_k[I_c]} \tag{42}$$

where: $\bar{r} = \frac{\sum \mathbf{r}[I_c]}{N_c}$. N_c is the number of points in set I_c.
$\mathbf{r}_k[I_g] = \mathbf{r}_k[I_g] - \bar{r}$.
$\mathbf{x}_k$ = gap or pressure.

$$\beta_{k+1} = \delta\,(G_{k+1}/G_k) \tag{43}$$

where: $G = \mathbf{x}_k[I_c]^T\mathbf{x}_k[I_c]$.
δ : If no change in contact area then $\delta = 0$ else $\delta = 1$.

$$\mathbf{d}_k[I_g] = \mathbf{r}_k[I_g] + \beta\mathbf{d}_k[I_g] \tag{44}$$

where: $\mathbf{d}_k$= descent direction.

The special feature here of the constrained conjugate gradient algorithm presented by Polonsky & Keer is that, it works in two ways, as the steepest descent when the contact area is changing and as a conjugate gradient when the contact area no longer changes.

3.1.2 Constrained Conjugate Gradient by INSA Lyon Group

The algorithm described by Bugnicourt et al. 2018 [2] (INSA Lyon Group) showed an outstanding performance in the *contact mechanics challenge* [11].

The algorithm developed by Insa Lyon Group, or later also referred as BUG, solves both the primal objective (Equation (17)) and the dual objective (Equation (21)), with or without adhesion (Section 2.1.2). In the original research paper, they are presented as two different algorithms IN and $OUT - LIN$, for solving the dual and primal problem, respectively. There is also a third algorithm called OUT discussed in the original paper, which has special functionality to handle the *mean pressure* constraint while solving the primal problem.

In this work, only one implementation is made that works with both the primal and the dual problem and has the same performance as presented in Bugnicourt et al. 2018 [2] BUG (discussed in the Results section later).

The constraints are directly applied on $\mathbf{x}_k$ and $\mathbf{r}_k$ such that all the points in contact, i.e., $\mathbf{x}_k \leq 0$ are pushed to $\mathbf{x}_k = 0$. Then the active set $\neg$ *mask bound* (logical NOT of *mask bound*) is determined. The points in ($\mathbf{x}_k \leq 0 \wedge$ (logical AND) $\mathbf{r}_k \geq 0$) are in *mask bound*. The values of $\mathbf{r}_k$ that are in active set (values in *mask bound*) are set to 0. Same is done for the descent direction.

On comparison with the standard conjugate gradient algorithm, the parameters can be defined as per Equation (45), (46), & (47).

$$\alpha_k = \frac{\mathbf{r}_k^T \mathbf{d}_k}{\mathbf{d}_k^T \mathbf{H} \mathbf{d}_k} \tag{45}$$

$$\beta_{k+1} = -\frac{\mathbf{r}_{k+1}^T \mathbf{H} \mathbf{d}_k}{\mathbf{d}_k^T \mathbf{H} \mathbf{d}_k} \approx -\frac{\mathbf{r}_{k+1}^T (\mathbf{r}_{k+1} - \mathbf{r}_k)}{\alpha_k \mathbf{d}_k^T \mathbf{H} \mathbf{d}_k} \tag{46}$$

$$\mathbf{d}_{k+1} = \mathbf{r}_{k+1} + \beta \mathbf{d}_k \tag{47}$$

In Equation (46), for β_{k+1}, the small simplification made using the value of α_k in the expression is done assuming that the problem is linear only for the computation of β_{k+1}. Then, $\mathbf{r}_{k+1} = \mathbf{r}_k - \alpha \mathbf{d}_k$ can be used to obtain the expression.

3.1.3 GFMD

The GFMD (Green's Function Molecular Dynamics) solver is described in Prodanov et al. 2013 [5]. GFMD integrates the equation of motion of the Fourier displacements $\tilde{u}$ with $\mathbf{m}(q)\ddot{\tilde{\mathbf{u}}} = Forces(\tilde{\mathbf{u}}, \dot{\tilde{\mathbf{u}}})$ in order to find the static equilibrium. It tries to solve Equation (48), making the interpretation that the system can be modeled as a system of damped harmonic oscillators. The Fourier modes (the pixels of $\tilde{u}$) are the harmonic oscillators. The equations of motion are integrated using the Verlet algorithm. Damping is considered such that the slowest mode is critically damped.

$$\begin{aligned} \tilde{\mathbf{f}}_{k+1} &= \tilde{\mathbf{f}}_k + \eta\{\tilde{\mathbf{u}}_k - \tilde{\mathbf{u}}_{k-1}\} \\ \tilde{\mathbf{u}}_{k+1} &= 2\tilde{\mathbf{u}}_k - \tilde{\mathbf{u}}_{k-1} + \tilde{\mathbf{f}}_{k+1}\Delta t^2 \end{aligned} \tag{48}$$

where: $\tilde{\mathbf{u}}$= gap in Fourier space.

$\tilde{\mathbf{f}}$ = force in Fourier space.

η= damping coefficient.

3.1.4 FIRE

Fast Inertial Relaxation Engine (FIRE) is a simple optimization algorithm originally used to optimize local atomic structure by Bitzek et al. 2006 [6], and further adapted

by Zhou et al. 2019 [3].

The algorithm aims to minimize the potential energy of the system by dynamically updating the time step and velocity of the system during the time propagation of GFMD's equation of motion, Equation (48). It works on the analogy of a skier going downhill on a slope.

The basic idea behind FIRE is that there is an internal averaging of gradient direction happening by assigning inertia to the variables. Since true dynamics do not matter in this case, the *instantaneous velocity* is made to slightly bias towards the steepest direction using Equation (49). The descent direction is kept in check by tracking $P = \mathbf{f}^T\mathbf{v}$ (if $P < 0$ i.e., direction changes, then, v, ζ & Δt are reset to initial values and if, $P > 0$ then, the system is accelerated in the same direction). It makes sure the direction of skier is always a descent direction.

$$\begin{aligned} \tilde{\mathbf{x}}_{k+1} &= \tilde{\mathbf{x}}_k + \tilde{\mathbf{v}}_k \Delta t + 0.5\tilde{\mathbf{f}}_k \Delta t^2 \\ \tilde{\mathbf{v}}_{k+1} &= (1-\zeta)\mathbf{v}_k + \zeta \left(\tilde{\mathbf{f}}_k / ||\tilde{\mathbf{f}}_k||\right) |\tilde{\mathbf{v}}_k| \end{aligned} \tag{49}$$

where: $\Delta t =$ time step
ζ = mixing factor
v = $\dot{x}$, velocity
$||.||=$ is the Frobenius norm

Listed are the tuning paramteres for the FIRE algorithm.

N_{min} = minimum molecular dynamics steps before accelerating the dynamics.
f_{inc} = factor to increment the time step Δt.
f_{dec} = factor to decrease the time step Δt.
f_{ζ} = factor to increment ζ.

3.1.5 Mass Weighted FIRE

Mass Weighted FIRE (MW-FIRE) was introduced in the paper by Zhou et al. 2019 [3], Algorithm (7) in the appendix. It uses the standard FIRE [6] methodology to find the minimum of the system with an additional aspect of 'mass' also playing a role in the minimization process. The mass is tuned such that the eigenfrequencies of individual harmonic oscillators are the same. The mass-weighted system of equations has a unit stiffness matrix.

The contact problem formulated in Section 2.2.1, is transformed into equation of motion in Equation (50).

$$m(q_l)\ddot{\tilde{u}}_l = \frac{q_l E^*}{2}\tilde{u}_l + F_{lj}\phi'(F_{ji}^{-1}(\tilde{u} - \tilde{h})_i) \tag{50}$$

The algorithm is called mass weighted since a non-unit mass is chosen such that, $m(q) \propto qE^*$,

$$\ddot{\tilde{u}}_l = \tilde{u}_l + \frac{1}{m(q_l)}F_{lj}\phi'(F_{ji}^{-1}(\tilde{u}_i - \tilde{h}_i)) \tag{51}$$

3.1.6 L-BFGS-B

Limited-memory BFGS for bound constraints (L-BFGS-B) is a quasi-Newton method that constructs an approximate Hessian using gradient history. It uses line search such that the step length α satisfies *Wolfe conditions*.

Scipy [12] implementation of the Limited-memory BFGS bounded algorithm is considered in this work. This implementation is based upon the L-BFGS algorithm described in the Nocedal [1].

In this work, L-BFGS-B is used to solve the constrained problem described in Section 2.2.2 and the unbound version i.e., Limited-memory BFGS algorithm (L-BFGS),

is used for solving the unconstrained problem described in Section 2.2.1. L-BFGS is the same algorithm as L-BFGS-B but with bounds deactivated.

3.2 Simulation System Setup

In this section, the physical parameters needed to describe a simulation system using a rough surface are discussed.

A simulation system is defined using three main parameters: elasticity, interaction and topography. Elasticity and interaction have already been discussed in Section 2.1.1 and Section 2.1.2, respectively. Self-affine random surfaces are considered in this work and these can be statistically defined by, the root-mean-square (RMS) slope (h'_{rms}), λ_l (long cutoff wavelength), λ_s (short cutoff wavelength), Hurst exponent (H) and physical sizes s. All of which are calculated using a power spectral density (PSD) of a surface [9, 21, 22, 23, 24]. In Figure 9(b), PSD of the surface with the same statistical parameters as described in Bugnicourt et al. 2018 [2] is shown.

The RMS height h_{rms}, RMS slope h'_{rms} and RMS curvature h''_{rms} are computed using the formulations described in the work of Jacobs et al. 2017 [21],

$$h_{rms}^2 = \frac{1}{A}\int_A h^2(x,y)dxdy \tag{52}$$

$$(h'_{rms})^2 = \frac{1}{A}\int_A |\nabla h(x,y)|^2 dxdy \tag{53}$$

$$(h''_{rms})^2 = \frac{1}{4A}\int_A |\nabla^2 h(x,y)|^2 dxdy \tag{54}$$

where: $A = L_x L_y$

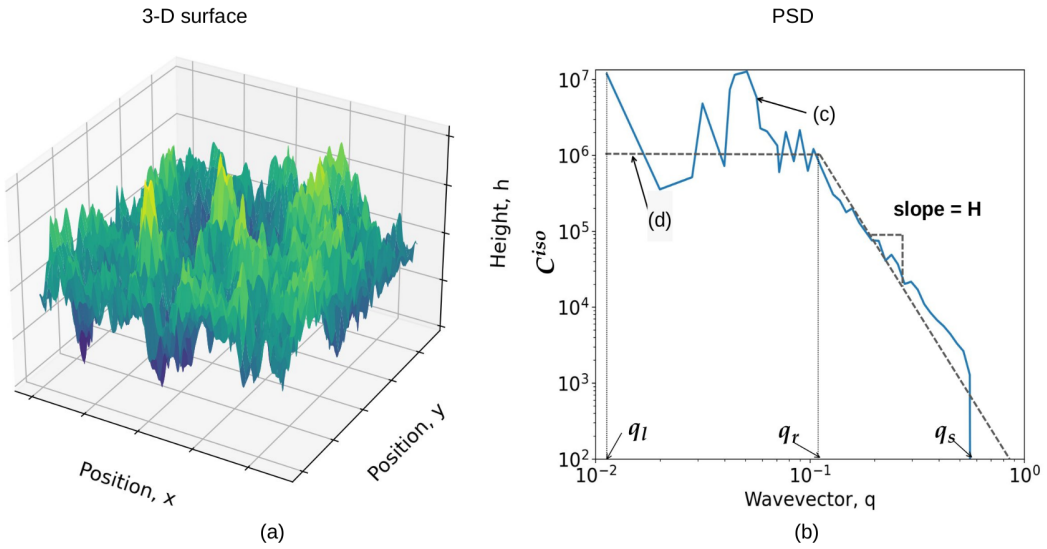


Figure 9: (a) 3-D rough surface as described in Bugnicourt et al. 2018 [2] in non-dimensional units. (b) Power spectral density of: (c) surface described in Bugnicourt et al. 2018 [2] and (d) an ideal PSD with no noises. q_l is the low wave-vector, q_s is the high wave-vector and q_r is the roll-off wave-vector.

q_s is the high-wavevector computed from the short-wavelength ($\frac{2\pi}{\lambda_s}$) cutoff of the power-law and $q_l = \frac{2\pi}{\lambda_l}$ is the low wavevector of the power-law. Figure 9(b) shows these regions on x-axis of the plot.

The Hurst exponent (H) is related to the exponent of the power law in the self-affine region of the PSD. Given by, $C^{iso}(q) \propto q^{-2-2H}$. Statistically same surface roughness is achieved by rescaling the heights by ξ^H, where ξ is the lateral scaling factor of q (wavevector) and $C^{iso}(q)$ is power-spectra for the self affine surface [21]. Typical values for Hurst exponent range from 0.7 to 0.9. In Figure 9, $H = 0.8$ is considered.

The physical size (L_x, L_y) of the topography is given by the product of the number of grid points and the pixel size.

4 Results

This section provides results to answer the following questions,

- What is the fastest constrained optimization algorithm for contact problems?
    - Is working with Primal problem better than Dual?
- What is the fastest unconstrained optimization algorithm for contact problems?
- Which algorithm is robust against instabilities?

4.1 Solving the Constrained Problem

For solving the constrained problem, hard-wall constraints are implemented inside the minimizing algorithm. Implementations of algorithms Polonsky & Keer CG, Algorithm (3) in the appendix, and INSA Lyon Group CG, Algorithm (4) in the appendix, are validated against the performances in the *ContactMechanics* python package [25] [26] and in Bugnicourt et al. 2018 [2]. These are discussed in Section 4.1.1.1 and Section 4.1.2.1, respectively. The Validation of INSA Lyon Group CG is shown in a later subsection after the results from the primal versus dual problem formulations using strongly adhesive system for its benchmark.

For verifying these algorithms' convergences, $\log(Error)$, i.e., log value of max absolute projected gradient versus iterations and fraction contact area change versus

iterations, are plotted. While, for the study for primal problem versus dual problem behavior, the log value of relative energy decay (i.e., $\log\left(\frac{E_i - E_{reference}}{E_{reference}}\right)$) is plotted with respect to number of iterations. The reason for this is that the gradients from the primal problem and the dual problem are two different quantities and cannot be compared directly, however, the final energy at the solution is the same up to the fifth or sixth decimal place. Plots with both primal and dual energies as reference are made to avoid any bias towards the reference energy system (e.g., the decay of primal energy with respect to the primal energy as a reference). The log value of relative fraction contact areas (i.e., $\log\left(\frac{\text{frac ar}_i - \text{frac ar}_{reference}}{\text{frac ar}_{reference}}\right)$) are plotted against the number of iterations. Here, the reference areas are from the respective primal and dual solutions to show clear convergence behavior with the two objectives.

4.1.1 Purely Elastic System

A purely elastic system is defined in (Equation (17)) & (Equation (21)).

The results for comparing the performance with primal objective (Equation (17) using, *OUT-LIN* & *OUT* algorithms) and dual objective (Equation (21) using *IN* algorithm) from the paper published by Bugnicourt et al. [2] are discussed for different contact areas in Figure 10.

4.1.1.1 Validating Polonsky & Keer

The PK implementation used in this work is validated against the implementation of the same algorithm in the Python package Contact Mechanics [26, 25]. The results of these tests are presented in Figure 11, where, the convergence and the change of fraction area are compared for the two. The simulation system parameters used for making this validation test are presented in Table 1.

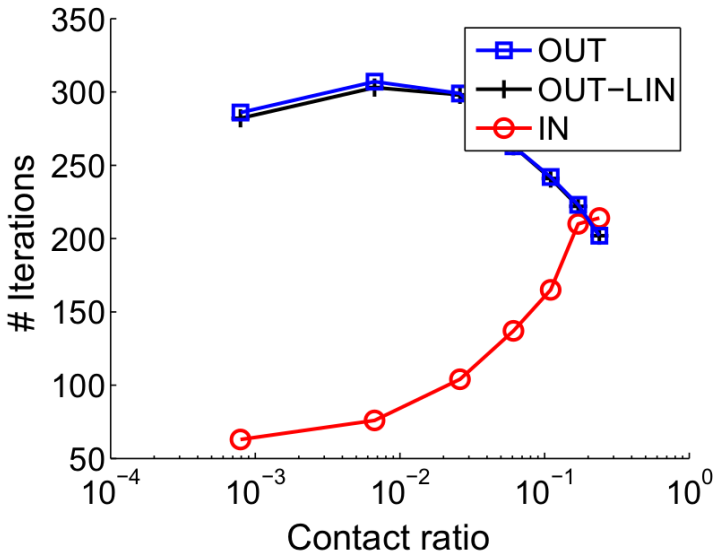


Figure 10: Contact area (or contact ratio) versus number of iterations (# Iterations). Reprinted from Bugnicourt et al. 2018 paper [2].

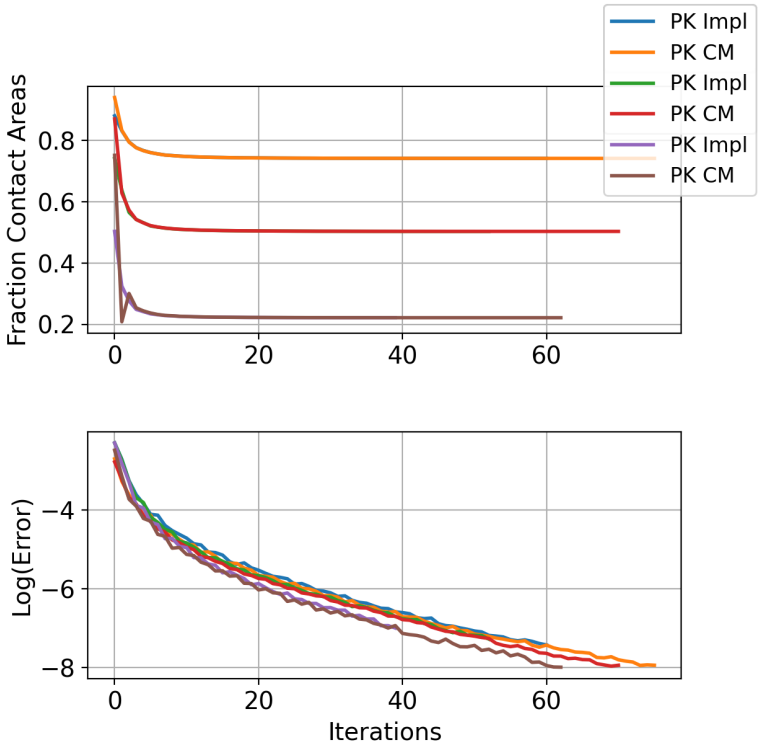


Figure 11: Comparing the convergences from PK implementation against CM package implementation.

Simulation system	
Parameter Name	Value
Dimension	2-dimensional
Number of grid points	(1024,1024)
Physical sizes	(1,1)
Elastic modulus E^*	50
Surface type	spherical
Radius	10

Table 1: Simulation system values for validating PK.

4.1.1.2 For Contact Area > 90% . pressure > 0

A rough contact system is used to compare performance from the primal problem and the dual problem. The parameters for the system used are defined in Table 2.

Simulation system	
Parameter Name	Value
Dimension	2-dimensional
Number of grid points	(1024,1024)
Physical sizes	(1,1)
Elastic modulus E^*	1.0
Surface type	random rough
RMS slope	1
Short cutoff wavelength λ_s	0.2
Long cutoff wavelength λ_l	0.5
Hurst exponent (H)	0.8

Table 2: Simulation system values for comparing the primal and dual problem.

In figures (12, 13 and 14), the first two curves show the decay of energy versus number of function evaluations, first, with respect to the energy from the dual objective and then, with respect to the energy from the primal objective. The third plot shows the change of the contact area with iterations.

In figure 15, the three algorithms in consideration are compared together in order to find the best performing algorithm for our test case of a non-adhesive test problem.

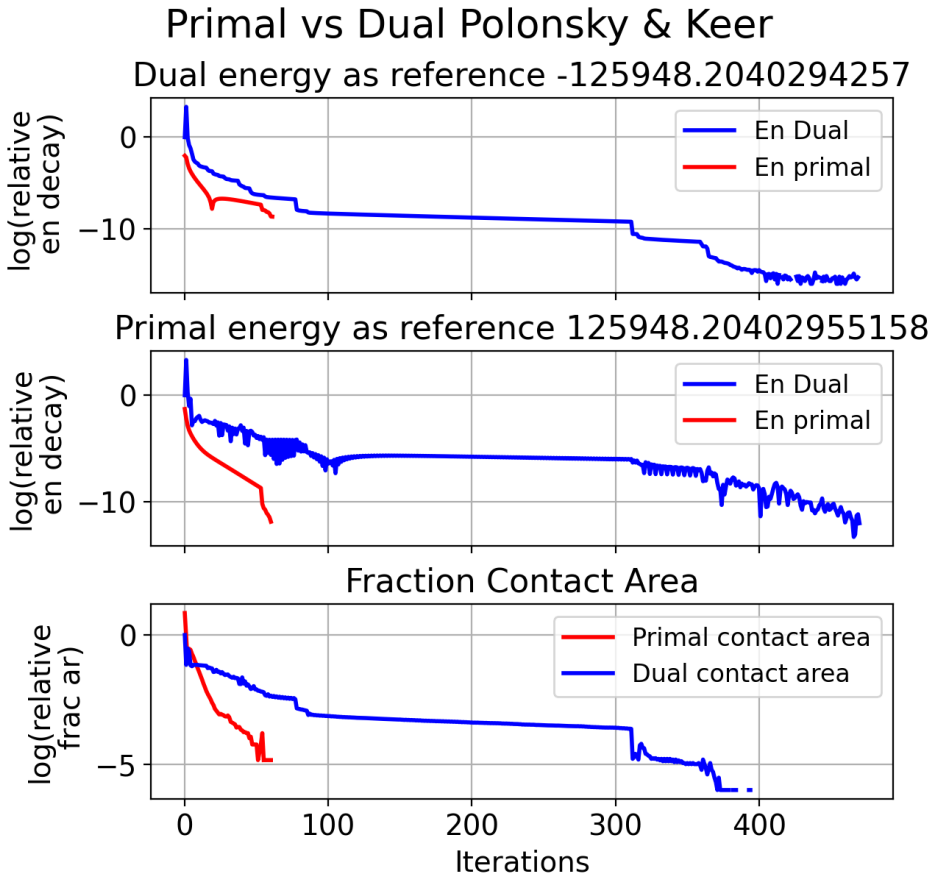


Figure 12: Decay in the energy of system and the change of fraction contact area with respect to number of iterations when using PK (Section 3.1.1).

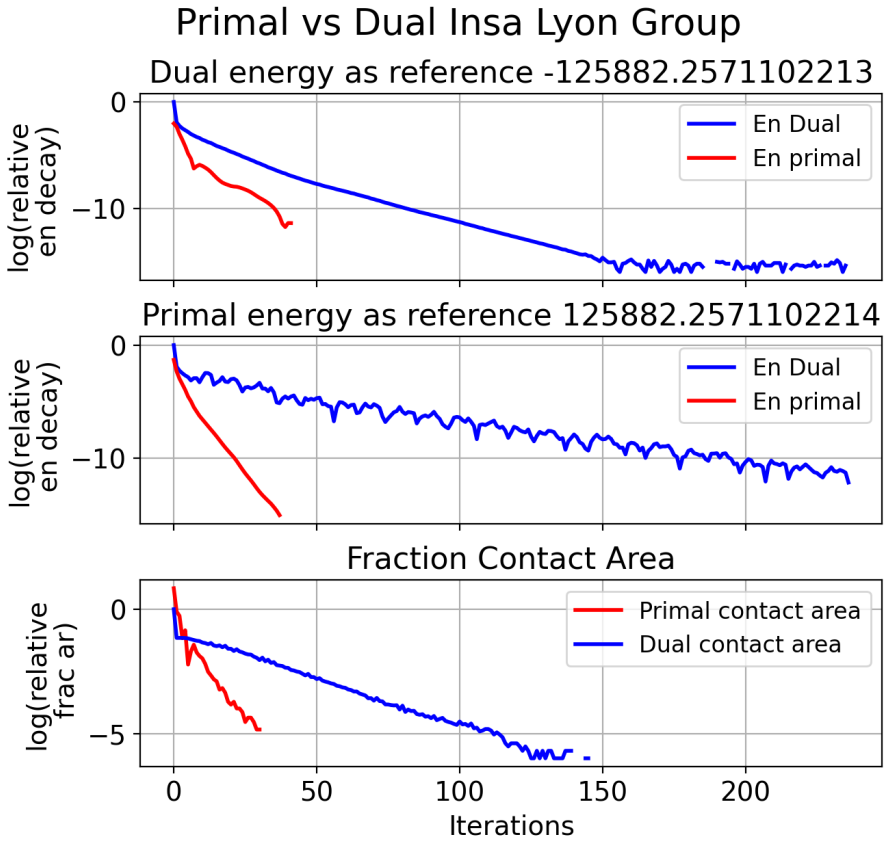


Figure 13: Decay in the energy of system and the change of fraction contact area with respect to number of iterations when using BUG (Section 3.1.2) algorithm.

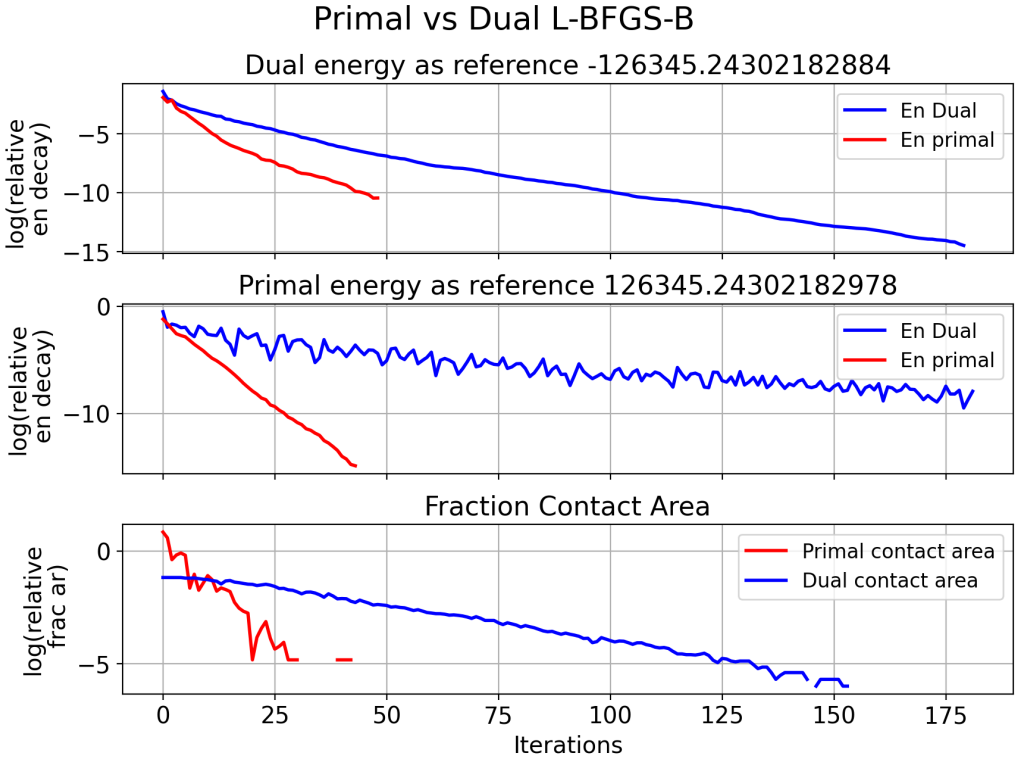


Figure 14: Decay in the energy of system and the change of fraction contact area with respect to number of iterations when using L-BFGS-B (Section 3.1.6) algorithm.

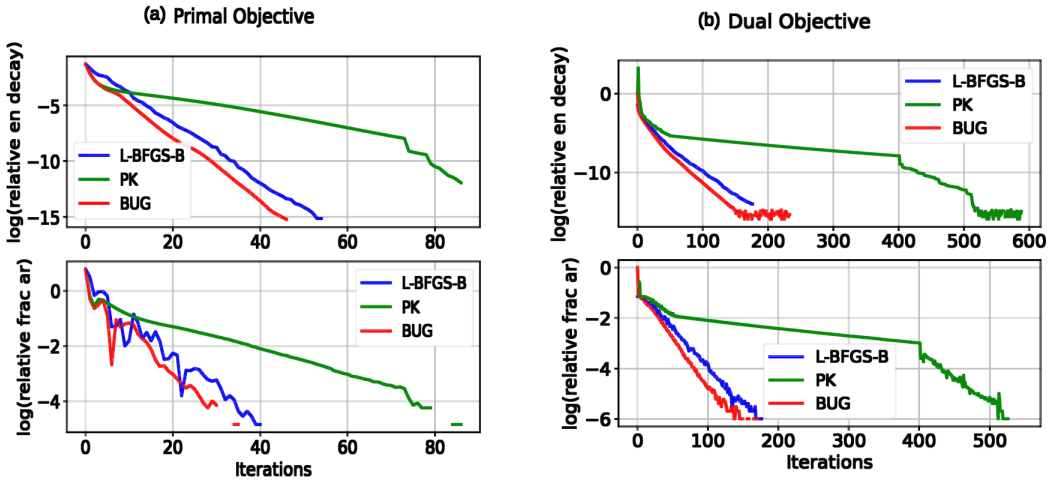


Figure 15: Decay in the energy of system and the change of fraction contact area with respect to number of iterations, for contact area $> 90\%$ solving, (a) primal objective and (b) dual objective.

4.1.1.3 For Contact Area $\simeq$ 10% . pressure > 0

Exactly the same system as defined in Table 2 is used here.

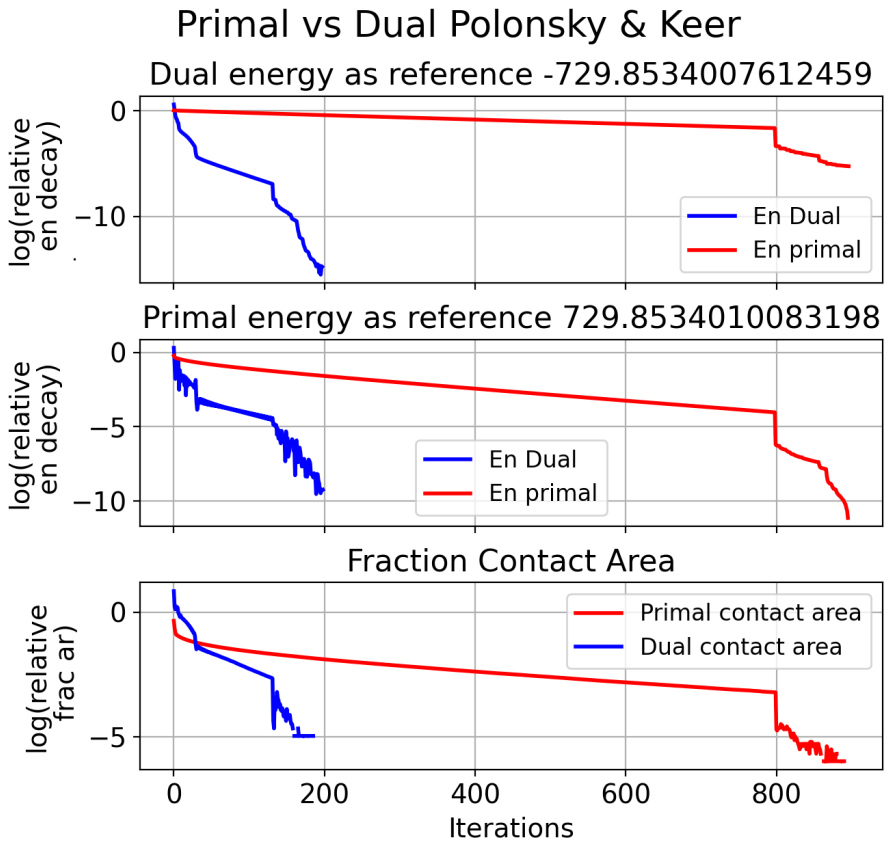


Figure 16: Decay in the energy of system and the change of fraction contact area with respect to number of iterations when using PK (Section 3.1.1) algorithm.

In Figures (16, 17 and 18), the first two curves show the decay of energy versus number of function evaluations, first, with respect to energy from dual objective and then, with respect to energy from primal objective. The third plot shows the way the contact area changes with iterations.

In Figures (19), the three algorithms in consideration are compared together to observe their performances for our test case of an elastic problem.

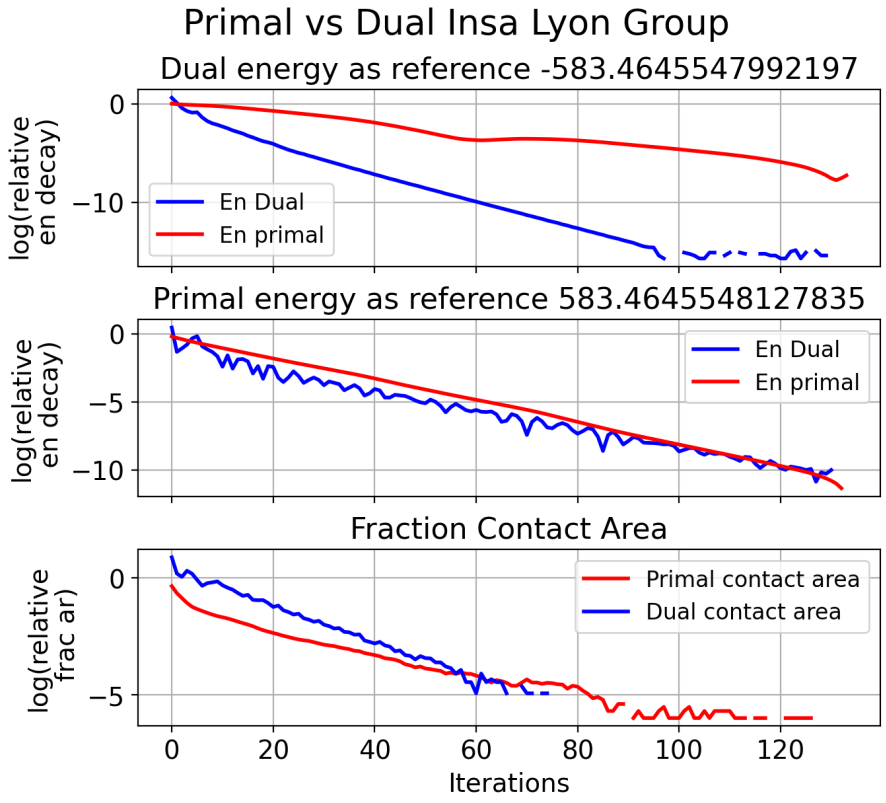


Figure 17: Decay in the energy of system and the change of fraction contact area with respect to number of iterations when using BUG (Section 3.1.2) algorithm.

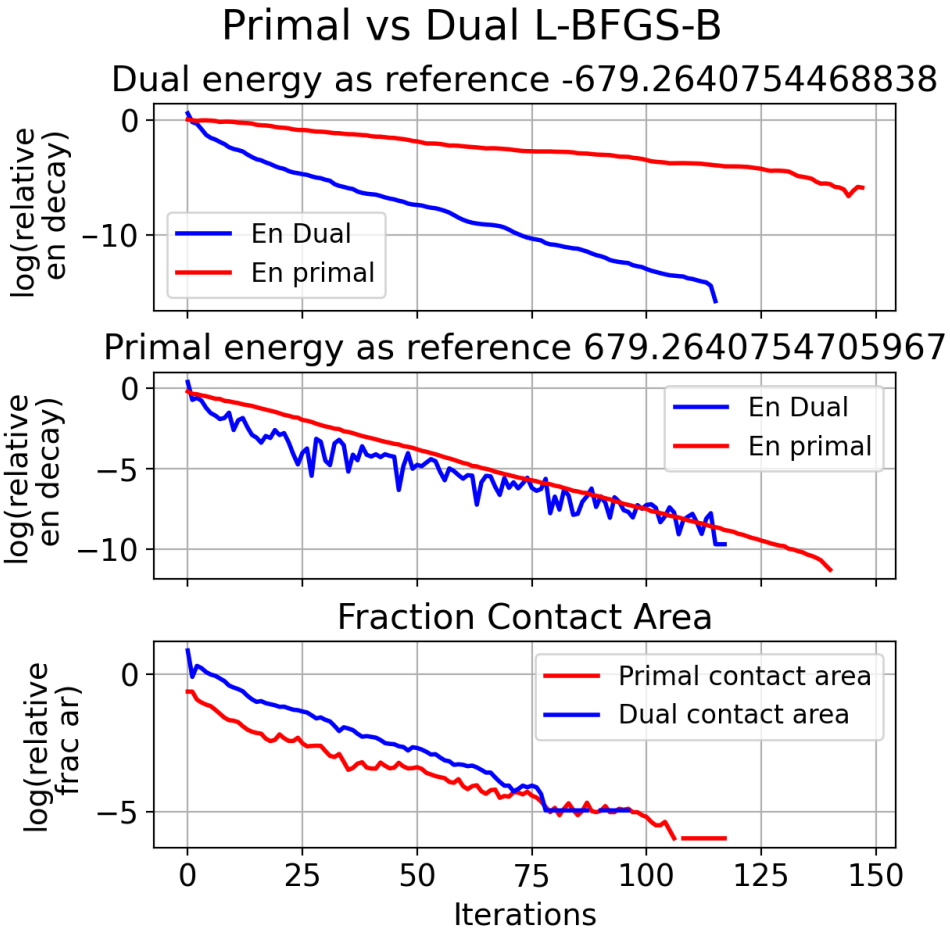


Figure 18: Decay in the energy of system and the change of fraction contact area with respect to number of iterations when using scipy L-BFGS-B (Section 3.1.6) algorithm.

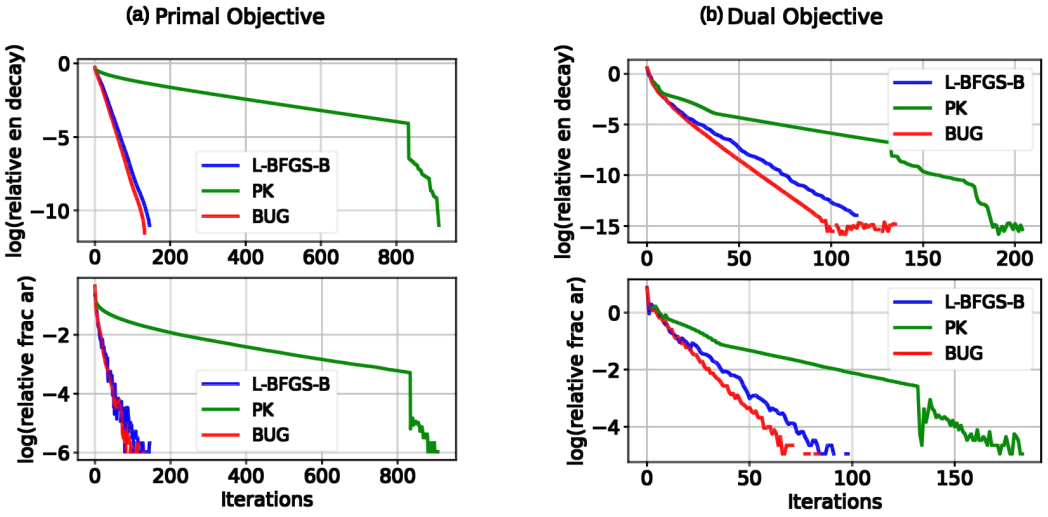


Figure 19: Decay in the energy of system and the change of fraction contact area with respect to number of iterations, for $\simeq 10\%$ contact area solving, (a) primal objective and (b) dual objective.

4.1.2 With Strong Adhesion

A system is said to be strongly adhesive when the tabor coefficient $\mu > 3.0$. In this test case, a strongly adhesive system from BUG paper [2] is considered. The parameters of the simulation setup are defined in the Table 3.

Simulation system	
Parameter Name	Value
Dimension	2-dimensional
Number of grid points	(2048,2048)
Physical sizes (s)	4μm
Elastic modulus E^*	25 MPa
Surface type	random rough
Short cutoff wavelength λ_s	$0.1\mu m$
Long cutoff wavelength λ_l	$1\mu m$
Hurst exponent (H)	0.8

Table 3: Simulation system values for validating the BUG algorithm.

The dual problem with strong adhesion is not tested in this work because it has been already tested by BUG [2] as shown in Figure 20. From the figure, it can be

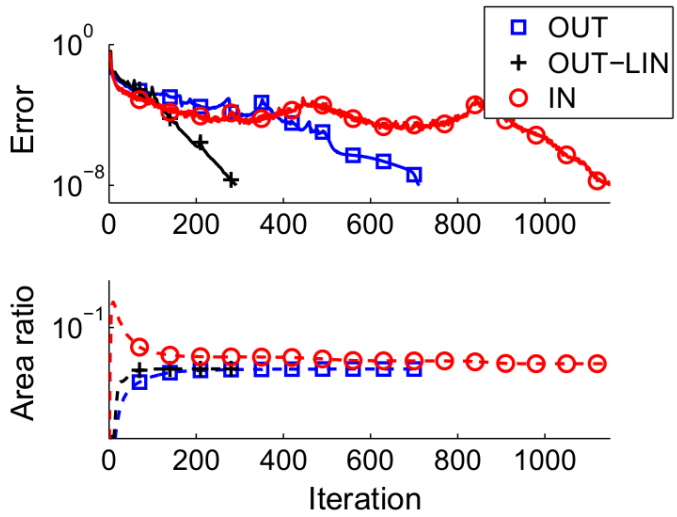


Figure 20: Gradients and contact area. Reprinted from Bugnicourt et al. [2] fig. 3.

noted that the performance of the algorithm(*IN* algorithm) working with the dual objective is worse as compared to the algorithms working with primal objective(*OUT* & *OUT-LIN*). More or less, it can be said that solving the dual problem does not work at all with strong adhesion.

4.1.2.1 Validating Insa Lyon Group (BUG) algorithm

In Figure 21, the implementation of BUG algorithm in this work is validated against the performance presented in Bugnicourt et al. 2018 [2] (in Figure 20).

4.1.2.2 Insa Lyon Group benchmark

In Figure 22, the performance of the three constrained solvers on the strongly adhesive problem (described in the paper by Bugnicourt et al. 2018 [2]) is tested. System presented in Table 3 is used in this test.

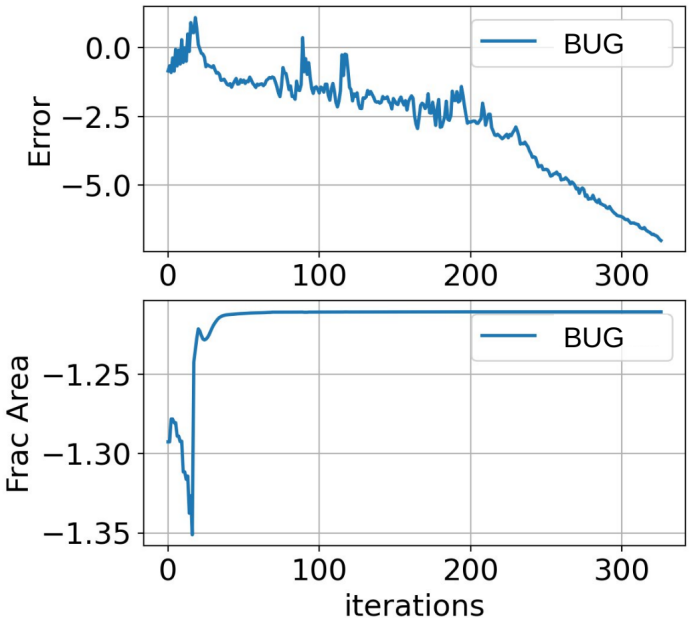


Figure 21: Performance (solving primal objective) of BUG implementation used in this work , originally called as OUT-LIN algorithm from Bugnicourt et al. 2018 [2].

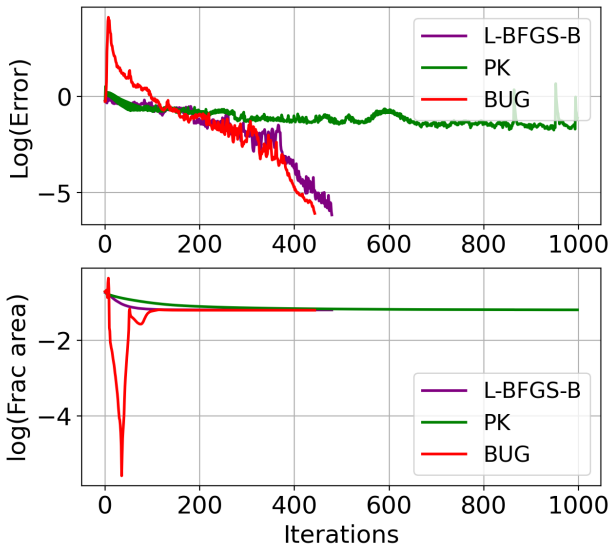


Figure 22: Decay in the log(error) of the system and change of log(contact area) with respect to number of iterations, for approximately 6% contact area.

4.2 Solving the Unconstrained Problem

The unconstrained problem formulated in Section 2.2.1 is now preconditioned in order to improve the performance of the solvers. The unconstrained problem under consideration is defined in the Fourier space. In order to discuss the implementation of preconditioning to the contact problem, first the optimisation in Fourier space is discussed in the next section and then the preconditioning technique.

4.2.1 Optimization in Fourier Space

For solving the minimisation problem in Fourier space, $\tilde{\mathbf{u}}$ needs to be defined in a way that optimising algorithms like L-BFGS can work with it. The reason for this is that L-BFGS needs N linearly independent float variables to work with, while in any general complex array they are $2N$ linearly dependent floats. However because of the hermitian symmetries N of them are linearly independent ($\tilde{\mathbf{u}} = (x_0, x_1 + iy_1, \ldots, x_1 - iy_1)$). This problem is overcome by rearranging the values inside $\tilde{\mathbf{u}}$ such that the new unknowns are now $x_0, x_1, x_2, \ldots, y_1$.

4.2.2 Preconditioning Applied to our Problem

The preconditioning technique chosen is motivated by the Mass weighting technique introduced by Zhou et al. 2019 [3].

The problem is formulated in the Fourier space because the stiffness matrix is diagonal in Fourier space. Diagonalizing the stiffness matrix $\mathbf{K}$ is not enough to accelerate the minimization process. Convergence speed was observed to decrease upon moving to the Fourier space. To accelerate the minimization, another substitution is done by updating $\tilde{\mathbf{u}}$ by $\tilde{\mathbf{v}} = \sqrt{\tilde{\mathbf{K}}}\tilde{\mathbf{u}}$ which leads to a condition number of 1 for the stiffness

matrix. Similar preconditioning techniques are discussed in works like Jiang et al. 2004 [27], Gergelits et al. 2018 [28] etc.

Only the elastic contribution to the Hessian is used ($\mathbf{L} = \sqrt{\tilde{\mathbf{K}}}$) because the stiffness matrix $\tilde{\mathbf{K}}$ remains constant over the iterations and it is also the dominant part of the Hessian in the adhesive problem.

The mathematical derivation for the preconditioning process is discussed in the following sections.

Preconditioning the Elastic Energy

Elastic energy is defined as,

$$E(\mathbf{u}) = \frac{1}{2} u_i K_{ij} u_j \tag{55}$$

and using Parseval's theorem we can write energy in terms of Fourier displacements as,

$$E(\tilde{\mathbf{u}}) = \frac{1}{2N} \tilde{u_i} \tilde{K_{ii}} \tilde{u_i} \tag{56}$$

The expression for energy, using symmetries of $\tilde{\mathbf{u}}$, $E(\tilde{\mathbf{u}})$ is written as,

$$E(\tilde{\mathbf{u}}) = \frac{1}{2N} \left(x_0^2 \tilde{K_{00}} + 2 \sum_{i}^{n//2+1} |x_i^2 + y_i^2| \tilde{K_{ii}} \right) \quad \text{for N odd and} \tag{57}$$

$$E(\tilde{\mathbf{u}}) = \frac{1}{2N}\left(x_0^2\tilde{K_{00}} + 2\sum_{i}^{n//2}|x_i^2 + y_i^2|\tilde{K_{ii}} + x_{n//2+1}^2\tilde{K}_{n//2+1,n//2+1}\right) \quad \text{for N even} \tag{58}$$

Considering $N = 6$, $\tilde{\mathbf{u}}$ is written as, $x_0,\ x_1 + iy_1,\ x_2 + iy_2,\ x_3,\ x_2 - iy_2,\ x_1 - iy_1$. The partial derivative of the energy along these elements or force can be defined as,

$$\begin{aligned}
\frac{\partial E}{\partial x_0} &= \frac{1}{2N}\left(2x_0\tilde{K}_{00}\right) = \frac{x_0 K_{00}}{N} \\
\frac{\partial E}{\partial x_1} &= \frac{1}{2N}\left(2(2x_1\tilde{K}_{11})\right) = \frac{2x_1\tilde{K}_{11}}{N} \\
\frac{\partial E}{\partial x_2} &= \frac{1}{2N}\left(2(2x_2\tilde{K}_{22})\right) = \frac{2x_2\tilde{K}_{22}}{N} \\
\frac{\partial E}{\partial x_3} &= \frac{1}{2N}\left(2x_3\tilde{K}_{33}\right) = \frac{x_3\tilde{K}_{33}}{N} \\
\frac{\partial E}{\partial y_2} &= \frac{1}{2N}\left(2(2y_2\tilde{K}_{22})\right) = \frac{2y_2\tilde{K}_{22}}{N} \\
\frac{\partial E}{\partial y_1} &= \frac{1}{2N}\left(2(2y_1\tilde{K}_{11})\right) = \frac{2y_1\tilde{K}_{11}}{N}
\end{aligned} \tag{59}$$

For odd 'N'(Equation (57)), the gradient of the energy with respect to the independent spectral degrees of freedom $(x_0, x_1, \ldots, y_1)$ is,

$$\frac{\partial E}{\partial(x_0, x_1, \ldots, y_1)} = \left(\frac{x_0\tilde{K}_{00}}{N}, \frac{2x_1\tilde{K}_{11}}{N}, \frac{2x_2\tilde{K}_{22}}{N}, \ldots, \frac{2y_2\tilde{K}_{22}}{N}, \frac{2y_1\tilde{K}_{11}}{N}\right) \tag{60}$$

and when the energy is considered for even number of grid points from Equation (58), then the force or the gradients can be written as,

$$\frac{\partial E}{\partial(x_0, x_1, \ldots, x_{n//2+1}, \ldots, y_1)} = \left(\frac{x_0 \tilde{K}_{00}}{N}, \frac{2x_1 \tilde{K}_{11}}{N}, \frac{2x_2 \tilde{K}_{22}}{N}, \ldots, \right.$$
$$\left. \frac{x_{n//2+1} \tilde{K}_{n//2+1,n//2+1}}{N}, \ldots, \frac{2y_2 \tilde{K}_{22}}{N}, \frac{2y_1 \tilde{K}_{11}}{N} \right) \tag{61}$$

After writing the equations for energy and force in Fourier space, displacement $\tilde{u}$ is then scaled by introducing $\tilde{\mathbf{v}} = \sqrt{\widetilde{\mathbf{K}}}\tilde{\mathbf{u}}$. The elastic energy defined in Equation (57) is re-defined as in Equation (62),

$$\begin{aligned} E &= \frac{1}{2N} \sum_i \tilde{u_i} \tilde{K}_{ii} \tilde{u_i} \\ &= \frac{1}{2N} \sum_i \tilde{u_i} \sqrt{\tilde{K}_{ii}} \sqrt{\tilde{K}_{ii}} \tilde{u_i} \\ &= \frac{1}{2N} \sum_i \tilde{v_i} \tilde{v_i} \end{aligned} \tag{62}$$

$$\tilde{\mathbf{u}} = (x_0, x_1 + iy_1, x_2 + iy_2, \ldots, x_2 - iy_2, x_1 - iy_1)$$

$$\tilde{\mathbf{v}} = \left(x_0 \sqrt{\tilde{K}_{00}}, \sqrt{\tilde{K}_{11}}(x_1 + iy_1), \sqrt{\tilde{K}_{22}}(x_2 + iy_2), \ldots, \right.$$
$$\left. \sqrt{\tilde{K}_{22}}(x_2 - iy_2), \sqrt{\tilde{K}_{11}}(x_1 - iy_1) \right)$$

$$\tilde{\mathbf{v}} = \left(x'_0, x'_1 + iy'_1, x'_2 + iy'_2, \ldots, x'_2 - iy'_2, x'_1 - iy'_1)\right)$$

where: $x'_i = \sqrt{\tilde{K}_{ii}} x_i$

$y'_i = \sqrt{\tilde{K}_{ii}} y_i$

then the new objective looks like Equation (63),

$$E = \frac{1}{2N} \left((x'_0)^2 + 2 \sum_{i=1}^{n//2+1} |(x'_i)^2 + (y'_i)^2| \right) \tag{63}$$

and the gradient for the preconditioned objective is defined as in Equation (64),

$$\frac{\partial E}{\partial (x'_0, x'_1, \ldots, y'_1)} = \left(\frac{x'_0}{N}, \frac{2x'_1}{N}, \frac{2x'_2}{N}, \ldots, \frac{2y'_2}{N}, \frac{2y'_1}{N} \right) \tag{64}$$

Energy from Equation (58) i.e., for even number of grid points is redefined as in Equation (65),

$$\tilde{\mathbf{u}} = (x_0, x_1 + iy_1, x_2 + iy_2, \ldots, x_{n//2+1}, \ldots, x_2 - iy_2, x_1 - iy_1)$$

$$\tilde{\mathbf{v}} = \Big(x_0 \sqrt{\tilde{K}_{00}}, \sqrt{\tilde{K}_{11}}(x_1 + iy_1), \sqrt{\tilde{K}_{22}}(x_2 + iy_2), \ldots, \sqrt{\tilde{K}_{n//2+1,n//2+1}} x_{n//2+1},$$
$$\ldots, \sqrt{\tilde{K}_{22}}(x_2 - iy_2), \sqrt{\tilde{K}_{11}}(x_1 - iy_1) \Big)$$

$$\tilde{\mathbf{v}} = \Big(x'_0, x'_1 + iy'_1, x'_2 + iy'_2, \ldots, x'_{n//2+1}, \ldots, x'_2 - iy'_2, x'_1 - iy'_1) \Big)$$

where: $x'_i = \sqrt{\tilde{K}_{ii}} x_i$.
$y'_i = \sqrt{\tilde{K}_{ii}} y_i$.

$$E = \frac{1}{2N}\left((x'_0)^2 + 2\sum_{i=1}^{n//2} |(x'_i)^2 + (y'_i)^2| + (x'_{n//2+1})^2\right) \tag{65}$$

and the force for the energy in Equation (65) is defined as in Equation (66),

$$\frac{\partial E}{\partial(x'_0, x'_1, \ldots, x'_{n//2+1}, \ldots, y'_1)} = \left(\frac{x'_0}{N}, \frac{2x'_1}{N}, \frac{2x'_2}{N}, \ldots, \frac{x'_{n//2+1}}{N}, \ldots, \frac{2y'_2}{N}, \frac{2y'_1}{N}\right) \tag{66}$$

Preconditioning the Adhesive Energy

Adhesive energy is defined as a function of gap. As shown in Equation (67).

$$E_{adh} = \sum_i \phi(F_{ij}^{-1} \tilde{g}_j) \tag{67}$$

where, $\tilde{\mathbf{g}}$ can be written as, $x_0,\ x_1 + iy_1,\ x_2 + iy_2, \ldots,\ x_2 - iy_2,\ x_1 - iy_1$.

Similar to the above derivation for elastic energy, same steps are repeated for calculating the expression for the gradient or force from the adhesive energy.

Considering $N = 6$, $\tilde{\mathbf{g}}$ can be written as, $x_0,\ x_1 + iy_1,\ x_2 + iy_2,\ x_3,\ x_2 - iy_2,\ x_1 - iy_1$. Then, the partial derivative of the energy along these elements or force can be defined as equations in (Equation (68)),

$$
\begin{aligned}
\frac{\partial E}{\partial x_0} &= \frac{\partial}{\partial x_0}\left(\sum_i \phi(F_{ij}^{-1}\tilde{g_j})\right) = \frac{1}{N}\sum_i \phi'(F_{ij}^{-1}\tilde{g_j})F_{i0}^{-1} \\
\frac{\partial E}{\partial x_1} &= \frac{\partial}{\partial x_1}\left(\sum_i \phi(F_{ij}^{-1}\tilde{g_j})\right) = \frac{2}{N}\sum_i \phi'(F_{ij}^{-1}\tilde{g_j})(F_{i1}^{-1}+F_{i5}^{-1}) \\
&= \frac{2}{N}\sum_i \phi'(F_{ij}^{-1}\tilde{g_j})cos(qx_1) \\
\frac{\partial E}{\partial x_2} &= \frac{\partial}{\partial x_2}\left(\sum_i \phi(F_{ij}^{-1}\tilde{g_j})\right) = \frac{2}{N}\sum_i \phi'(F_{ij}^{-1}\tilde{g_j})(F_{i2}^{-1}+F_{i4}^{-1}) \\
&= \frac{2}{N}\sum_i \phi'(F_{ij}^{-1}\tilde{g_j})cos(qx_2) \\
\frac{\partial E}{\partial x_3} &= \frac{\partial}{\partial x_3}\left(\sum_i \phi(F_{ij}^{-1}\tilde{g_j})\right) = \frac{1}{N}\sum_i \phi'(F_{ij}^{-1}\tilde{g_j})F_{i3}^{-1} \\
\frac{\partial E}{\partial y_2} &= \frac{\partial}{\partial y_2}\left(\sum_i \phi(F_{ij}^{-1}\tilde{g_j})\right) = \frac{2\sum_i \phi'(F_{ij}^{-1}\tilde{g_j})(iF_{i2}^{-1}-iF_{i4}^{-1})}{N} \\
&= -\frac{2}{N}\sum_i \phi'(F_{ij}^{-1}\tilde{g_j})sin(qx_2) \\
\frac{\partial E}{\partial y_1} &= \frac{\partial}{\partial y_1}\left(\sum_i \phi(F_{ij}^{-1}\tilde{g_j})\right) = \frac{2\sum_i \phi'(F_{ij}^{-1}\tilde{g_j})(iF_{i1}^{-1}-iF_{i5}^{-1})}{N} \\
&= -\frac{2}{N}\sum_i \phi'(F_{ij}^{-1}\tilde{g_j})sin(qx_1)
\end{aligned}
\tag{68}
$$

where: $F_{ij}^{-1}\tilde{g_j}$ = Fourier inverse of $\tilde{g}$ (gap in Fourier space).

ϕ = interaction potential.

N = grid size.

From the equations in Equation (68), for calculating the gradient or the force for a given $\tilde{\mathbf{g}}$, a generalized expression is derived (for odd number of grid points) in Equation (69),

$$\frac{\partial E}{\partial(x_0, x_1, \ldots, y_1)} = \left(\frac{1}{N} \sum_i \phi'(F_{ij}^{-1} \tilde{g}_j) F_{i0}^{-1}, \frac{2}{N} \sum_i \phi'(F_{ij}^{-1} \tilde{g}_j) cos(qx_1), \right. \\ \left. \ldots, -\frac{2}{N} \sum_i \phi'(F_{ij}^{-1} \tilde{g}_j) sin(qx_1) \right) \tag{69}$$

and when there are even number of grid points, the equation for force is defined as,

$$\frac{\partial E}{\partial(x_0, x_1, \ldots, x_{n//2+1}, \ldots, y_1)} = \left(\frac{1}{N} \sum_i \phi'(F_{ij}^{-1} \tilde{g}_j) F_{i0}^{-1}, \frac{2}{N} \sum_i \phi'(F_{ij}^{-1} \tilde{g}_j) cos(qx_1), \right. \\ \ldots, \frac{1}{N} \sum_i \phi'(F_{ij}^{-1} \tilde{g}_j) F_{i,n//2+1}^{-1}, \\ \left. \ldots, -\frac{2}{N} \sum_i \phi'(F_{ij}^{-1} \tilde{g}_j) sin(qx_1) \right) \tag{70}$$

The function for adhesive energy of the system is updated for the change of variable as, Equation (71),

$$\begin{aligned} E &= \phi(F_{ij}^{-1} \tilde{\tilde{g}}_j) \\ &= \phi(F_{ij}^{-1}(\tilde{u}_j - \tilde{h}_j)) \\ &= \phi\left(F_{ij}^{-1} \left(\frac{\tilde{v}_j}{\sqrt{\tilde{K}_j}} - \tilde{h}_j \right) \right) \end{aligned} \tag{71}$$

and the general expression for adhesive force becomes Equation (72), for odd number of grid points,

$$\frac{\partial E}{\partial(x'_0, x'_1, \ldots, y'_1)} = \left(\frac{1}{N\sqrt{\tilde{K}_0}} \sum_i \phi'(F_{ij}^{-1}\tilde{g}_j)F_{i0}^{-1}, \frac{2}{N\sqrt{\tilde{K}_1}} \sum_i \phi'(F_{ij}^{-1}\tilde{g}_j)cos(qx_1), \right.$$
$$\left. , \ldots, -\frac{2}{N\sqrt{\tilde{K}_1}} \sum_i \phi'(F_{ij}^{-1}\tilde{g}_j)sin(qx_1) \right) \tag{72}$$

and when there are even number of grid points then the adhesive force is defined as, in Equation (73),

$$\frac{\partial E}{\partial(x'_0, x'_1, \ldots, x'_{n//2+1}, \ldots, y'_1)} = \left(\frac{1}{N\sqrt{\tilde{K}_0}} \sum_i \phi'(F_{ij}^{-1}\tilde{g}_j)F_{i0}^{-1}, \frac{2}{N\sqrt{\tilde{K}_1}} \sum_i \phi'(F_{ij}^{-1}\tilde{g}_j)cos(qx_1), \right.$$
$$, \ldots, \frac{1}{N\sqrt{\tilde{K}_{n//2+1}}} \sum_i \phi'(F_{ij}^{-1}\tilde{g}_j)F_{in//2+1}^{-1},$$
$$\left. \ldots, -\frac{2}{N\sqrt{\tilde{K}_1}} \sum_i \phi'(F_{ij}^{-1}\tilde{g}_j)sin(qx_1) \right) \tag{73}$$

Performances when solving the unconstrained problem described as Section 2.2.1 and the preconditioned problem described in Section 4.2.2 are presented in the next sections. The potential is linearized for small gaps when using FIRE to avoid numerical overflow.

4.2.3 Validating Mass Weighted FIRE

The benchmark system under test here is a smooth system with a periodic substrate and a one dimensional random rough surface as defined in Table 4. Interaction length of the system is defined to be 2.56×10^{-4} and coefficient of attraction, $\gamma_{att} = 2.05 \times R_c$

(defined in Table 4) and coefficient of repulsion, $\gamma_{rep} = 2.10 \times 10^3 \times R_c$. The interaction potential defined in Equation (13) is used in this test. It is the same parameters as the test in Zhou et al. 2019 [3], but they also had lateral displacements in their system while here only normal displacements are considered.

Simulation system	
Parameter Name	Value
Dimension	1-dimensional
Number of grid points	(1024,)
Physical size (s)	(1024,)
Elastic modulus E^*	1.0
Surface type	random rough
radius of curvature (R_c)	$1/h''_{rms}$
Short cutoff wavelength λ_s	0.01s
Long cutoff wavelength λ_l	s
Hurst exponent (H)	0.8

Table 4: Simulation system values for comparing validating MW-FIRE.

The performance of the implementation of MW-FIRE considered in this work, Algorithm (7) in the appendix, is validated in Figure 23 against the performance described in the paper by Zhou et.al. 2019 [3], shown in Figure 24.

From Figure 24, it has to be noted that the original implementation of MW-FIRE takes only a little less than 200 iterations to converge up to energy of 10^{-5}. While, in the MW-FIRE implementation considered in this work decreases energy by 10^{-5} first in around 80 iterations and then, again in around 200 iterations (Figure 23).

4.2.4 Preconditioned L-BFGS Benchmark

4.2.4.1 Using System Defined in Section 4.2.3

The performance of preconditioned L-BFGS is presented in Figure 25 against the test system described in Section 4.2.3.

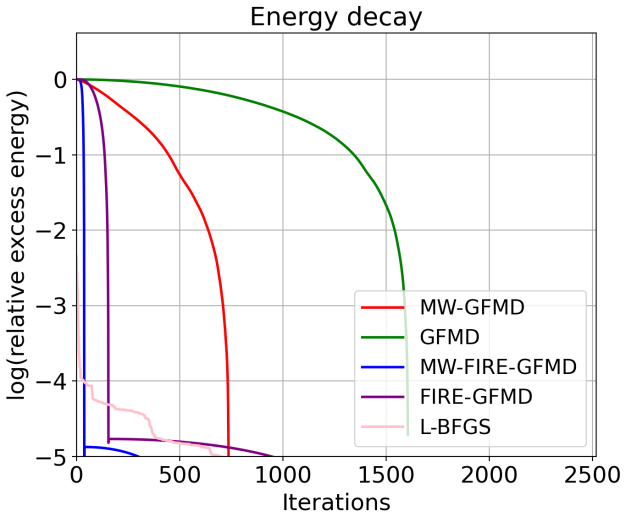


Figure 23: Decay in the energy of system with respect to number of iterations using MW-FIRE implementation used in this work, on the unconstrained system described in Zhou et al. 2019 paper [3].

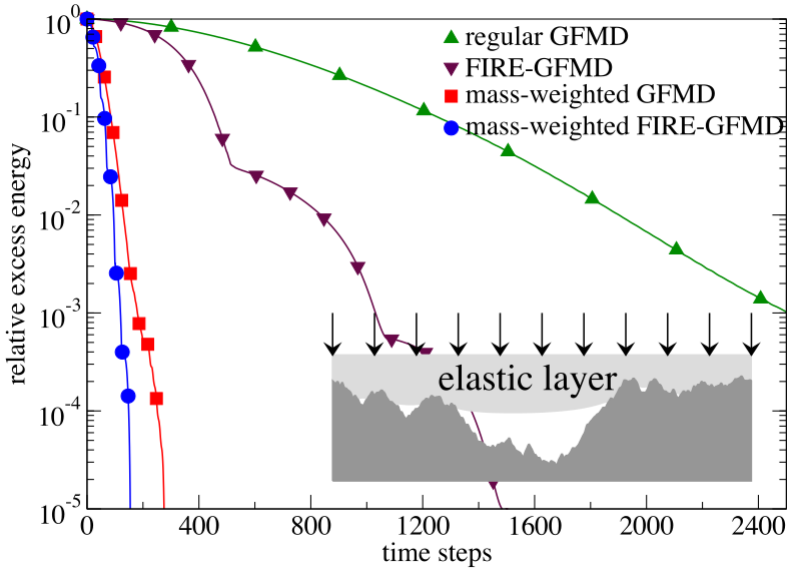


Figure 24: Decay in the energy of system with respect to number of iterations. Reprinted from Zhou et al. 2019 paper [3]

.

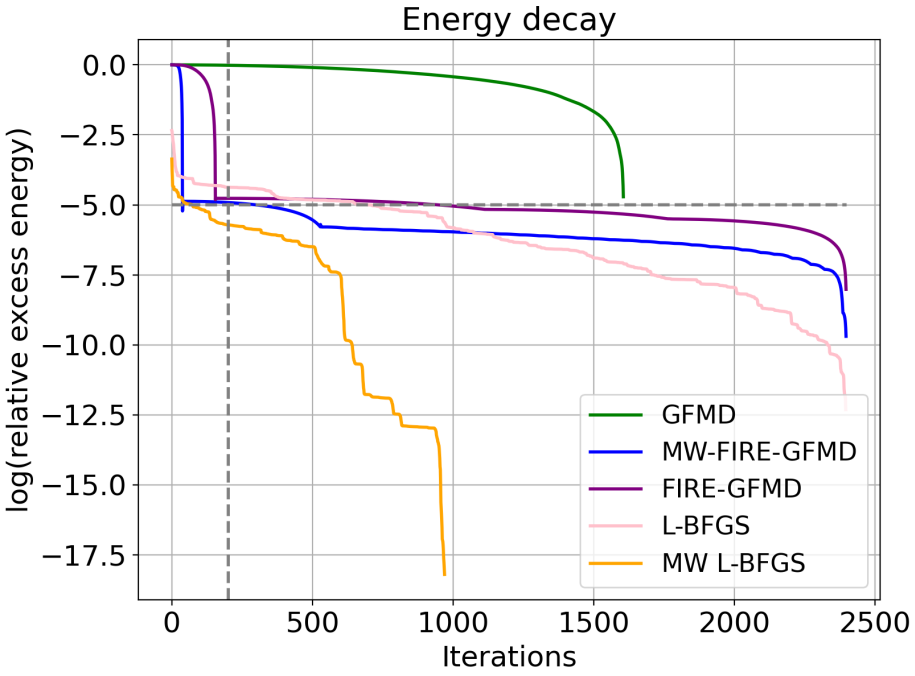


Figure 25: Decay in the energy of system with respect to number of iterations on system defined in Zhou et al. 2019 [3].

4.2.4.2 With 100 times higher Interaction range

To test the performance of preconditioned L-BFGS (or MW L-BFGS), the system from Section 4.2.3 was considered but with 100 times higher interaction range. Results from relative energy decay versus number of iterations and for gradients versus number of iterations are shown in Figure 26 and Figure 27, respectively. This was the best performance for MW-FIRE after tuning the parameters.

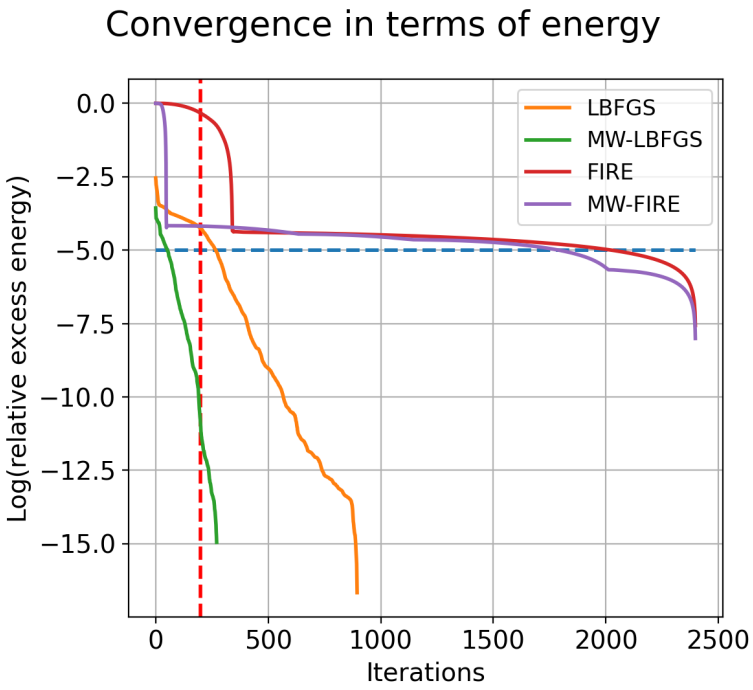


Figure 26: Decay in the energy of system with respect to number of iterations. Dashed lines show the performance for MW-FIRE from [3].

4.3 Robustness against Instabilities

In this section, an assessment is done to test whether the existing linear algorithms', such as BUG, still work on a system with instabilities. Also, the performances of the algorithms on constrained and unconstrained problem are compared.

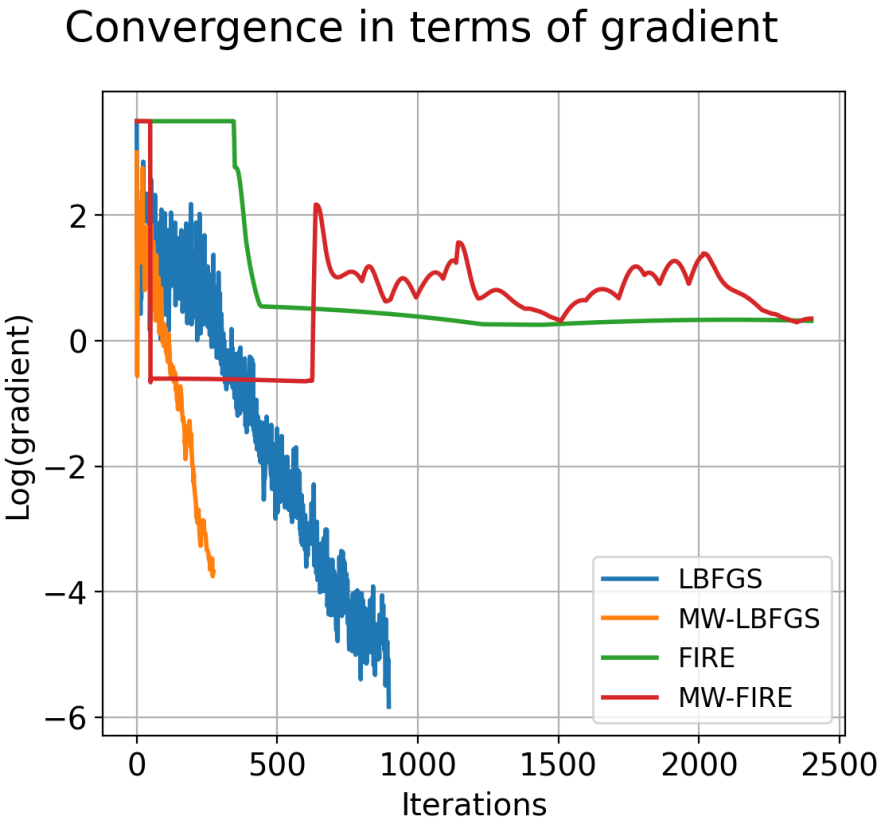


Figure 27: log of max absolute values of real space gradients with respect to number of iterations.

A "double sinewave" system shown in Figure 28, is considered with varying penetration values at intervals of 0.01, for a periodic substrate having interaction range of 0.0153. With Units h and λ being equal to 1 and composite elastic modulus $E^* = 1/\pi$. Table 5 presents the rest of the parameters needed to define the system.

Simulation system	
Parameter Name	Value
Dimension	1-dimensional
Number of grid points	(8192,)
Physical size (s)	(1,)
Elastic modulus E^*	$1.0/\pi$
Surface type	double sinewave
sinewave amplitude	0.05
sinewave number of periods	32

Table 5: Simulation system values for testing algorithms against instabilities.

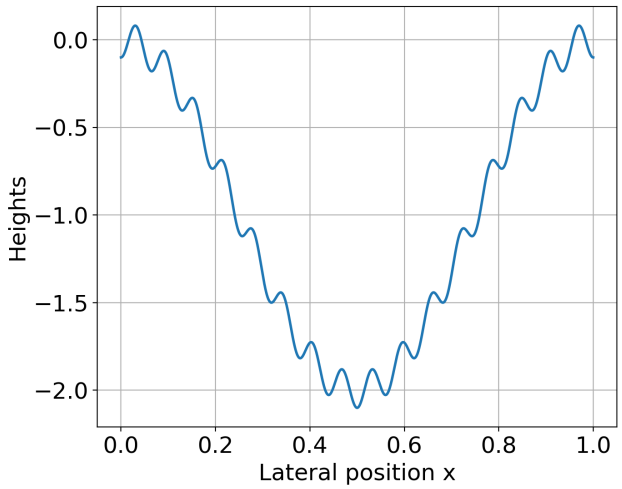


Figure 28: One dimensional topography profile with double sinewave instabilities.

Algorithms BUG and L-BFGS-B are used to solve the constrained minimization problem and preconditioned L-BFGS and a non-preconditioned L-BFGS is used to minimize the unconstrained problem.

To verify that the constrained and unconstrained systems are exactly the same and that there is no hysteresis due to discretization, Figure 29 and Figure 30 are plotted.

The hysteresis observed is a physical hysteresis only.

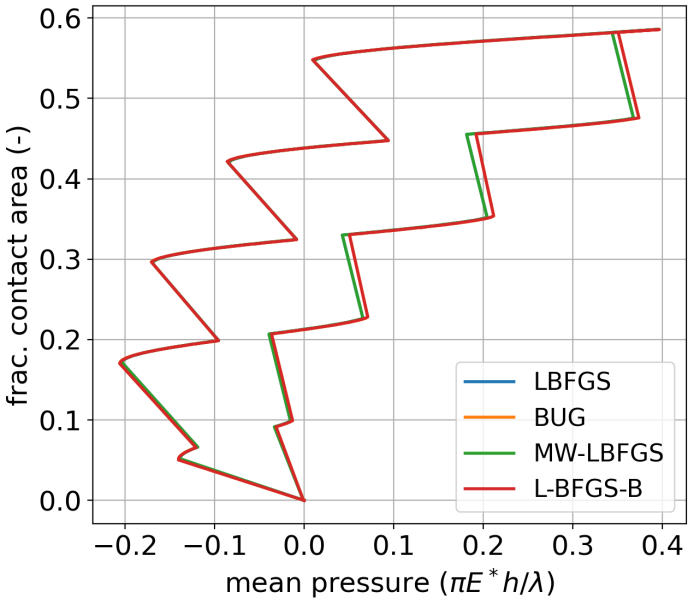


Figure 29: Penetration versus contact area to verify if there is any hysteresis.

In Figure 31 and Figure 32, performances of the algorithms are shown in terms of the number of function evaluations each algorithm does to converge to the minimum. The walltimes (in seconds) are also plotted to compare the time per iteration as it differs between the algorithms.

The constrained algorithms are also tested on a non-adhesive, one dimensional system shown in, Figure 33. The non-adhesive system is described in Table 6.

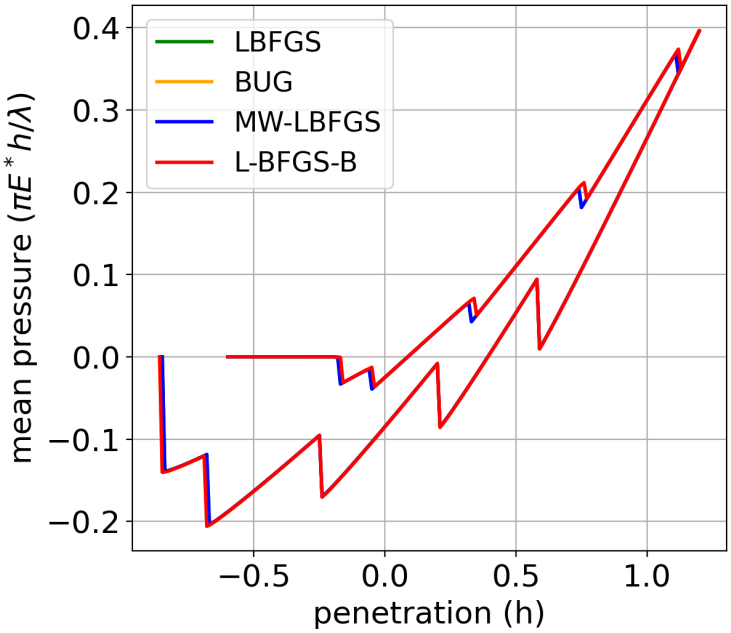


Figure 30: Penetration versus mean pressure to verify if there is any hysteresis.

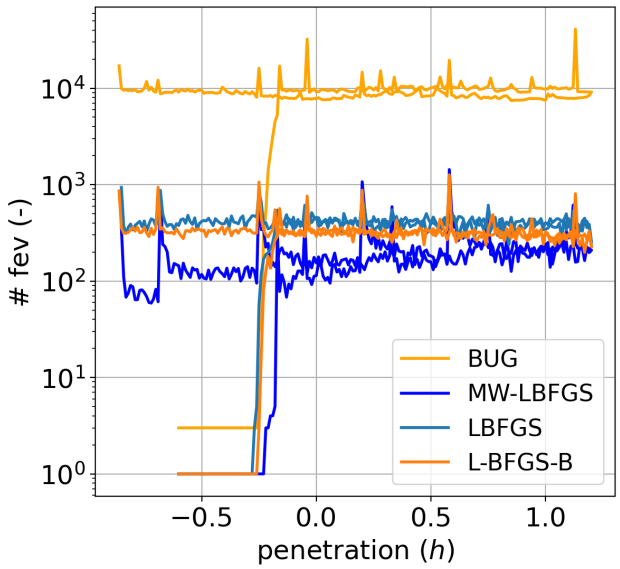


Figure 31: Penetration versus number of function evaluations (# fev) to observe the performances.

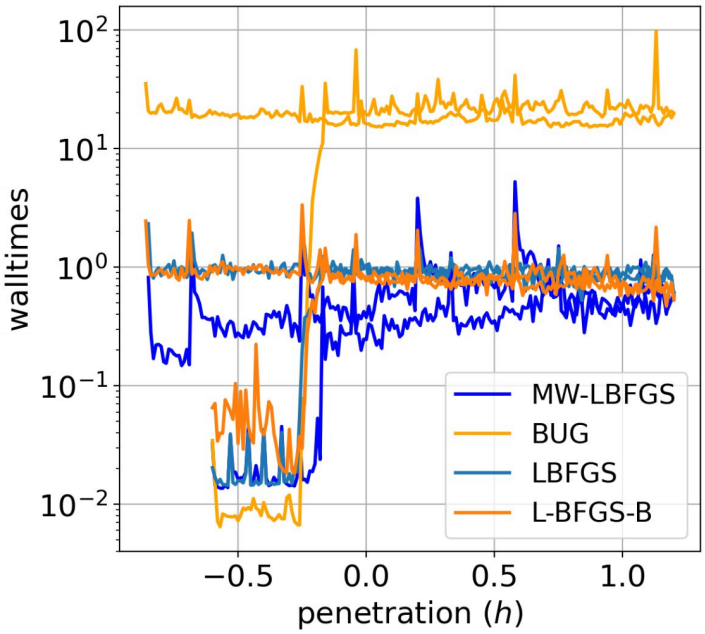


Figure 32: Penetration versus walltimes to observe the performances.

Simulation system	
Parameter Name	Value
Dimension	1-dimensional
Number of grid points	(2048,)
Physical size (s)	(1,)
Elastic modulus E^*	$1.0/\pi$
Surface type	double sinewave
sinewave amplitude	0
sinewave number of periods	16

Table 6: Simulation system values for non-adhesive 1-dimensional system to test BUG algorithm.

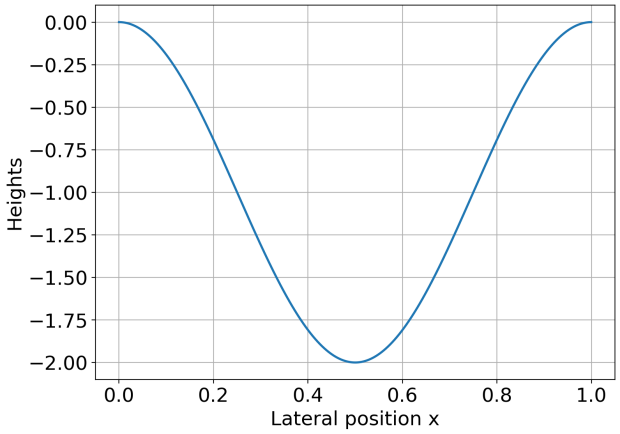


Figure 33: One dimensional topography profile with non-adhesive system.

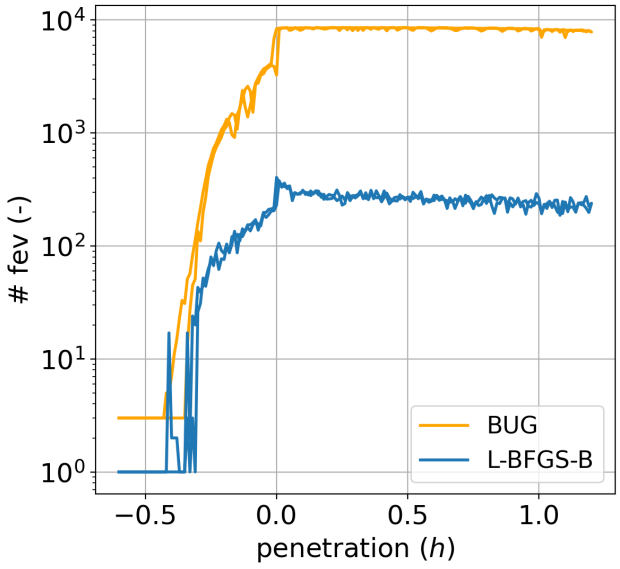


Figure 34: Penetration versus number of function evaluations (# fev) to observe the performances in the non-adhesive system .

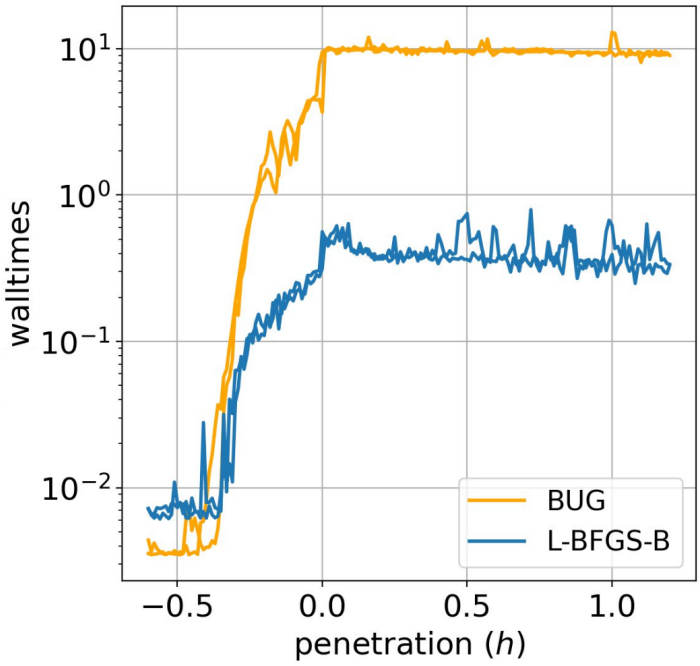


Figure 35: Penetration versus walltimes for the non-adhesive system.

5 Discussion

First, the behavior with constrained problem is studied.

Validation of CG algorithms: The implementation of Polonsky & Keer algorithm (PK) used is validated in Figure 11, where the convergence behaviour of the two PK is exactly the same. The implementation of BUG algorithm is also validated in Figure 21 against the results shown in Figure 20 from original Bugnicourt et al. 2018 [2] paper, where, the algorithm *OUT-LIN* takes nearly 330 iterations to converge to a primal contact area of roughly 6% when the mean value is controlled.

Primal vs. Dual: Upon validating the algorithms, the performances using the primal and the dual problems are tested for high and low contact areas, respectively. Before starting the discussion on results for our implementations, in Figure 10, the contact area versus the number of iterations are shown for BUG, to give, a priori knowledge on what is the expected behavior. As it can be seen, for very low contact areas, working with the dual problem is better than working with the primal problem. Looking at the intersection of these curves, one can say that this trend reverses the role i.e., primal works good for high contact areas.

For high($> 90\%$) contact area(pressure > 0) (Figure 15), performances when solving the primal problem are four to five times better for the BUG and the L-BFGS-B

algorithms and six times better for the PK algorithm, as compared to solving the dual problem.

For $\simeq 10\%$ contact area, the PK algorithm takes four times less iterations for solving the dual problem as compared to the primal problem, as expected. On contrary to our expectation, the BUG and the L-BFGS-B algorithms take nearly the same number of iterations, when solving both the primal and the dual problem, Figure 19.

Strong adhesion: In Figure 22 the performance of PK, BUG and L-BFGS-B is tested on a primal strongly adhesive system for approximately 6% contact area. And it is observed that PK does not converge on this problem while, the L-BFGS-B algorithm and the BUG algorithm take approximately same number of iterations to converge. Tests using strong adhesion while solving the dual problem are not done in the light of the fact, that the outcomes from the paper by Bugnicourt et al. 2018 [2] in Figure 20 clearly show that the performance is very poor when solving the dual problem with strong adhesion.

Studying the performance using unconstrained objective.

Validating MW-FIRE: First, the MW-FIRE implementation (Algorithm (7) in the appendix), is validated against the performance shown in the paper by Zhou et al. 2019 [3]. The results from the validation test are plotted in Figure 23, and Figure 24 presents the results from the original paper. L-BFGS solver is also added to compare its performance. It can be observed from the Figure 23 that, it takes first almost 100 and then, again 300 iterations for MW-FIRE to decrease relative excess energy to 10^{-5} while L-BFGS takes nearly 700 iterations to do the same. The performance for MW-FIRE is not the same as presented in the original paper because it has has a lot of tuning parameters and one needs some experience to tune them.

Preconditioned L-BFGS: Using observations from the MW-FIRE versus L-BFGS performance, the preconditioned objective formulated in Section 4.2.2 is used with L-BFGS solver to improve its performance. And the results from this are shown in Figure 25, where in 200 iterations *preconditioned L-BFGS* reduces the energy to 10^{-6}, while MW-FIRE reduces the energy only until 10^{-5} and L-BFGS could only manage to a factor of $10^{-4.5}$. After nearly 800 iterations, the energy is already reduced to a factor of 10^{-13} by preconditioned L-BFGS while, MW-FIRE just saturates at 10^{-5} and non-preconditioned L-BFGS could reduce till $10^{-5.}$ to 10^{-6}. To further test the performance of the preconditioned L-BFGS algorithm, the gradients and the decay of energy versus number of iterations were plotted in Figure 26 and Figure 27, respectively. It can be noted that, preconditioned L-BFGS outperforms all the solvers. One special thing to be noted in Figure 27 is that, the magnitude of gradients of FIRE and MW-FIRE never decreases. The reason for this might be the fact that the parameters for FIRE and MW-FIRE have not been tuned properly.

Handling instabilities: Now a comparison is done between solving the constrained problem and solving the unconstrained problem with instabilities in the system to test the robustness of the methods. To verify that the constrained and unconstrained formulations are equivalent, Figure 29 and Figure 30 are examined to check if there is any discretisation induced hysteresis or difference in the behavior of the two objectives and indeed the two are the same. The performances of the four solvers considered here namely, BUG and L-BFGS-B for constrained problem (since they showed a robust behavior in the results discussed in Figure 15 and Figure 19) and preconditioned L-BFGS and non-preconditioned L-BFGS for the unconstrained problem, are plotted in Figure 31 and Figure 32. MW-FIRE is not chosen because it takes a lot of time to tune the parameters.

It can be clearly seen in Figure 31 and Figure 32 that the preconditioned L-BFGS or *Mass Weighted L-BFGS* algorithm is the best performing solver among all the

solvers considered. The difference in time taken to find the minimum or the number of function evaluations between the L-BFGS solver for unconstrained problem and the L-BFGS-B solver for the constrained problem is very small.

Figure 34 and Figure 35 show that the BUG algorithm does not perform well because it is a one dimensional system. So the test is not conclusive on whether BUG is still performing well on two dimensional adhesive systems with instabilities.

6 Summary & Conclusion

To make the study of rough contact mechanics problems more efficient, robustness and easiness to solve the mathematical system of equations is very important. Until now, most of the benchmarks made to study the performances and robustness of these algorithms against each other were limited to a few specific problem statements. In this work the rough contact mechanics problem with adhesion is discussed in a much broader mathematical aspect, where the study starts by contrasting the primary and dual problems and then goes to constrained and unconstrained optimization problem. In the process a more efficient solver is also developed that outperforms all the solvers covered in this study such as, Constrained CG by Polonsky & Keer 2019 [4], Constrained CG by Bugnicourt et al. 2018 [2], GFMD (Green's Function Molecular Dynamics) from Prodanov et al. 2013 [5], FIRE (Fast Inertial Relaxation Engine) by Bitzek et al. 2006 [6], Mass weighted FIRE by Zhou et al. 2019 [3] and Scipy implementation of L-BFGS-B [12].

Using these works as background, in this work an attempt is made to answer three main questions:

- Which is the best algorithm for Constrained optimization problem ?
    - Is working with Primal problem better than Dual?
- Which is the best algorithm for Unconstrained optimization problem ?

- Which algorithms are robust against instabilities ?

To answer the first question, the constrained problem is formulated in two ways: as a primal problem and as a dual problem. The results in Section 4.1.1 clearly show that using primal objective for high contact area yields best performance. And from the Figures 15 and 19, it can be concluded that Insa Lyon Group constrained conjugate gradient [2], Algorithm (4) in the appendix, is the best solver among the test cases considered for non-adhesive problem.

Figure 22 shows that BUG is faster then L-BFGS-B and PK on a strongly adhesive system.

For answering the second question, reference work of Zhou et al. 2019 [3] is considered, wherein, the MW-FIRE algorithm is said to be the best performing algorithm for their test case. As part of this work, the new "Preconditioned L-BFGS" solver technique was developed for solving unconstrained problems and is tested against MW-FIRE in Figure 25. The performance using preconditioned L-BFGS is better than MW-FIRE, with less number of tuning parameters.

Until now, it can be said that for the test cases considered in this work, BUG algorithm is the best solver for the constrained problem and preconditioned L-BFGS, for the unconstrained problem. Now, these solvers were put against each other in a system with instabilities (Section 4.3) to test their robustness against negative Hessians.

From Figure 31 and Figure 32, it can be said that solving the unconstrained problem with preconditioned L-BFGS solver could be the more efficient technique, as it produces the best results for the considered test cases, even with instabilities on a one dimensional problem.

Due to the lack of time, the two dimensional case of Fourier L-BFGS could not be implemented. This would be the next step towards further improving this work. Applying preconditioning to conjugate gradient algorithms or trust-region algorithms

(since they are more robust in nature), could also be an interesting direction in which improvements can be made.

Bibliography

[1] J. Nocedal and S. J. Wright, *Numerical optimization*. Springer series in operations research, New York: Springer, 2nd ed ed., 2006. OCLC: ocm68629100.

[2] R. Bugnicourt, P. Sainsot, D. Dureisseix, C. Gauthier, and A. A. Lubrecht, "FFT-Based Methods for Solving a Rough Adhesive Contact: Description and Convergence Study," *Tribology Letters*, vol. 66, p. 29, Mar. 2018. 00000.

[3] Y. Zhou, M. Moseler, and M. H. Müser, "Solution of boundary-element problems using the fast-inertial-relaxation-engine method," *Physical Review B*, vol. 99, p. 144103, Apr. 2019. 00000.

[4] I. A. Polonsky and L. M. Keer, "A numerical method for solving rough contact problems based on the multi-level multi-summation and conjugate gradient techniques," *Wear*, vol. 231, pp. 206–219, July 1999.

[5] N. Prodanov, W. B. Dapp, and M. H. Müser, "On the contact area and mean gap of rough, elastic contacts: Dimensional analysis, numerical corrections, and reference data," *Tribology Letters*, vol. 53, pp. 433–448, Dec. 2013.

[6] E. Bitzek, P. Koskinen, F. Gähler, M. Moseler, and P. Gumbsch, "Structural Relaxation Made Simple," *Physical Review Letters*, vol. 97, p. 170201, Oct. 2006. 00000.

[7] B. Luan and M. Robbins, "The breakdown of continuum models for mechanical contacts. nature 435, 929-932," *Nature*, vol. 435, pp. 929–32, 07 2005.

[8] T. Jacobs, K. Ryan, P. Keating, D. Grierson, J. Lefever, K. Turner, J. Harrison, and R. Carpick, "The effect of atomic-scale roughness on the adhesion of nanoscale asperities: A combined simulation and experimental investigation," *Tribology Letters*, vol. 50, pp. 81–93, 04 2013.

[9] L. Pastewka and M. O. Robbins, "Contact between rough surfaces and a criterion for macroscopic adhesion," *Proceedings of the National Academy of Sciences*, vol. 111, no. 9, pp. 3298–3303, 2014.

[10] Z. Yu and H. Cheng, "Tunable adhesion for bio-integrated devices," *Micromachines*, vol. 9, p. 529, 10 2018.

[11] M. H. Müser, W. B. Dapp, R. Bugnicourt, P. Sainsot, N. Lesaffre, T. A. Lubrecht, B. N. J. Persson, K. Harris, A. Bennett, K. Schulze, S. Rohde, P. Ifju, W. Gregory Sawyer, T. Angelini, H. A. Esfahani, M. Kadkhodaei, S. Akbarzadeh, J.-J. Wu, G. Vorlaufer, A. Vernes, S. Solhjoo, A. I. Vakis, R. L. Jackson, Y. Xu, J. Streator, A. Rostami, D. Dini, S. Medina, G. Carbone, F. Bottiglione, L. Afferrante, J. Monti, L. Pastewka, M. O. Robbins, and J. A. Greenwood, "Meeting the Contact-Mechanics Challenge," *Tribology Letters*, vol. 65, p. 118, Dec. 2017.

[12] P. Virtanen, R. Gommers, T. E. Oliphant, M. Haberland, T. Reddy, D. Cournapeau, E. Burovski, P. Peterson, W. Weckesser, J. Bright, S. J. van der Walt, M. Brett, J. Wilson, K. J. Millman, N. Mayorov, A. R. J. Nelson, E. Jones, R. Kern, E. Larson, C. J. Carey, İ. Polat, Y. Feng, E. W. Moore, J. VanderPlas, D. Laxalde, J. Perktold, R. Cimrman, I. Henriksen, E. A. Quintero, C. R. Harris, A. M. Archibald, A. H. Ribeiro, F. Pedregosa, P. van Mulbregt, and SciPy 1.0 Contributors, "SciPy 1.0: Fundamental Algorithms for Scientific Computing in Python," *Nature Methods*, vol. 17, pp. 261–272, 2020.

[13] C. Campañá and M. H. Müser, "Practical green's function approach to the simulation of elastic semi-infinite solids," *Phys. Rev. B*, vol. 74, p. 075420, Aug 2006.

[14] L. Pastewka, T. A. Sharp, and M. O. Robbins, "Seamless elastic boundaries for atomistic calculations," *Phys. Rev. B*, vol. 86, p. 075459, Aug 2012.

[15] K. L. Johnson, *Contact Mechanics*. Cambridge University Press, 1985.

[16] L. Pastewka and M. Robbins, "Contact area of rough spheres: Large scale simulations and simple scaling laws," *Applied Physics Letters*, vol. 108, p. 221601, 05 2016.

[17] M. Müser, "Single-asperity contact mechanics with positive and negative work of adhesion: Influence of finite-range interactions and a continuum description for the squeeze-out of wetting fluids," *Beilstein journal of nanotechnology*, vol. 5, pp. 419–37, 04 2014.

[18] M. E. Fisher and B. Wiodm, "Decay of correlations in linear systems," *The Journal of Chemical Physics*, vol. 50, no. 9, pp. 3756–3772, 1969.

[19] H. Van der Vorst, "Iterative krylov methods for large linear systems," vol. 13, 04 2003.

[20] E. Vollebregt, "The bound-constrained conjugate gradient method for non-negative matrices," *Journal of Optimization Theory and Applications*, vol. 162, pp. 931–953, 2014.

[21] T. D. B. Jacobs, T. Junge, and L. Pastewka, "Quantitative characterization of surface topography using spectral analysis," *Surface Topography: Metrology and Properties*, vol. 5, p. 013001, jan 2017.

[22] M. H. Müser, "A dimensionless measure for adhesion and effects of the range of adhesion in contacts of nominally flat surfaces," *Tribology International*, vol. 100, pp. 41 – 47, 2016. 42nd Leeds-Lyon Symposium on Tribology- Surfaces and Interfaces: Mysteries at Different Scales.

[23] S. Hyun, L. Pei, J.-F. Molinari, and M. O. Robbins, "Finite-element analysis of contact between elastic self-affine surfaces," *Phys. Rev. E*, vol. 70, p. 026117, Aug 2004.

[24] C. Campañá, B. N. J. Persson, and M. H. Müser, "Transverse and normal interfacial stiffness of solids with randomly rough surfaces," *Journal of Physics: Condensed Matter*, vol. 23, p. 085001, feb 2011.

[25] L. Pastewka, T. Junge, and A. Sanner, "Python package: Contactmechanics," 2020. `https://github.com/ComputationalMechanics/ContactMechanics`.

[26] B. Weber, T. Suhina, T. Junge, L. Pastewka, A. Brouwer, and D. Bonn, "Molecular probes reveal deviations from amontons' law in multi-asperity frictional contacts," *Nature Communications*, vol. 9, 03 2018.

[27] L. Jiang, R. Byrd, E. Eskow, and R. Schnabel, "A preconditioned l-bfgs algorithm with application to molecular energy minimization," p. 17, 11 2004.

[28] T. Gergelits, K.-A. Mardal, B. Nielsen, and Z. Strakoš, "Laplacian preconditioning of elliptic pdes: Localization of the eigenvalues of the discretized operator," *SIAM Journal on Numerical Analysis*, vol. 57, pp. 1369–1394, 01 2019.

Appendices

Appendix

Algorithm 1 Conjugate direction method [1]

Initialise

Set initial $\mathbf{x}$ to any value.

Set conjugate directions, $\mathbf{d} = d_0, d_1, \ldots, d_{n-1}$.

while $\epsilon \leq gtol$ **do** :

$\alpha_k \leftarrow \dfrac{\mathbf{r}_k^T \mathbf{r}_k}{\mathbf{d}_k^T \mathbf{H} \mathbf{d}_k}$

$\mathbf{x}_{k+1} \leftarrow \mathbf{x}_k + \alpha_k \mathbf{d}_k$

$\mathbf{r}_{k+1} \leftarrow \mathbf{r}_k + \alpha_k \mathbf{H} \mathbf{d}_k$

$k \leftarrow k + 1$

end while

Algorithm 2 Standard Conjugate Gradient [1]

Initialise

Set initial $\mathbf{x}$ to any value.

Set $r_0 \leftarrow Hx_0 - b, p_0 \leftarrow -r_0, k \leftarrow 0;$

and for nonlinear $f_0 = f(x_0), \nabla f_0 = \nabla f(x_0); d_0 \leftarrow -\nabla f_0, k \leftarrow 0;$

while $\epsilon \leq gtol$ **do** :

$$\alpha_k \leftarrow \frac{\mathbf{r}_k^T \mathbf{r}_k}{\mathbf{d}_k^T \mathbf{H} \mathbf{d}_k}$$
$$\mathbf{x}_{k+1} \leftarrow \mathbf{x}_k + \alpha_k \mathbf{d}_k$$
$$\mathbf{r}_{k+1} \leftarrow \mathbf{r}_k + \alpha_k \mathbf{H} \mathbf{d}_k$$
$$\beta_{k+1} \leftarrow \frac{\mathbf{r}_{k+1}^T \mathbf{r}_{k+1}}{\mathbf{r}_k^T \mathbf{r}_k}$$
$$\mathbf{d}_{k+1} \leftarrow -\mathbf{r}_{k+1} + \beta_{k+1} \mathbf{d}_k$$
$$k \leftarrow k + 1$$

end while

Algorithm 3 Polonsky & Keer Implementation CCG [4]

Initialise $\mathbf{x}$

$\mathbf{m}ask_c = \mathbf{x} > 0$

$\mathbf{x}[\neg\mathbf{mask_c}] = 0$

$\mathbf{r}_k \leftarrow \nabla f(\mathbf{x}_k)$

if mean value **then**

$\mathbf{r}_k = \mathbf{r}_k - \overline{\mathbf{r}_k[\mathbf{mask_c}]}$,

end if

$R_k = \mathbf{r}_k[\mathbf{mask_c}]^T \mathbf{r}_k[\mathbf{mask_c}]$

$\mathbf{d}_k[\mathbf{mask_c}] = -\mathbf{r}_k[\mathbf{mask_c}]$

while $\epsilon \leq gtol$ **do** :

$\mathbf{h}_k = HessProd(\mathbf{d}_k)$

if mean value **then**

$\mathbf{h}_k = \mathbf{h}_k - \overline{\mathbf{h}_k[\mathbf{mask_c}]}$

end if

$\alpha_k = -\dfrac{\mathbf{r}_k[\mathbf{mask_c}]^T \mathbf{d}_k[\mathbf{mask_c}]}{\mathbf{h}_k[\mathbf{mask_c}]\mathbf{d}_k[\mathbf{mask_c}]}$

$\mathbf{x}_{k+1}[\mathbf{mask_c}] = \mathbf{x}_k[mask_c] + \alpha_k \mathbf{d}_k[\mathbf{mask_c}]$.

$\mathbf{mask_c} = \mathbf{x}_{k+1} > 0$

$\mathbf{x}_{k+1}[\neg\mathbf{mask_c}] = 0$

$\mathbf{mask_{bound}} = \mathbf{x}_{k+1} == 0 \; AND \; \mathbf{r}_k < 0$.

if $\mathbf{mask_{bound}} = \emptyset$ **then** $\delta = 1$

else $\delta = 0$ and $\mathbf{x}_{k+1}[\mathbf{mask_{bound}}] \leftarrow \mathbf{x}_{k+1}[\mathbf{mask_{bound}}] - \alpha_k \mathbf{r}_k[\mathbf{mask_{bound}}]$.

end if

if mean value **then**

$\mathbf{x}_{k+1} = (\text{mean value}/mean(\mathbf{x}_{k+1}))\mathbf{x}_{k+1}$

end if

$\mathbf{r}_{k+1} \leftarrow \nabla f(\mathbf{x}_k)$

if mean value **then**

$\mathbf{r}_{k+1} = \mathbf{r}_{k+1} - \overline{\mathbf{r}_{k+1}[\mathbf{mask_c}]},$

end if

$R_{k+1} = \mathbf{r}_{k+1}[\mathbf{mask_c}]^T \mathbf{r}_k[\mathbf{mask_c}]$

$\beta_{k+1} = \delta(R_{k+1}/R_k)$

$\mathbf{d}_{k+1}[\mathbf{mask_c}] = -\mathbf{r}_{k+1}[\mathbf{mask_c}] + \beta_{k+1}\mathbf{d}_k[\mathbf{mask_c}]$

$\mathbf{d}_{k+1}[\neg\mathbf{mask_c}] = 0$

$\epsilon = max(\mathbf{r}_{k+1}).$

end while

Algorithm 4 Insa Lyon Group CCG [2]

Initialise $\mathbf{x}$

descent direction, $\mathbf{d} = -\nabla f(\mathbf{x})$.

while $\epsilon \leq gtol$ **do** :

$\mathbf{h}_k = HessProd(\mathbf{d}_k)$

Compute $\alpha_k = -\dfrac{\mathbf{r}_k^T \mathbf{d}_k}{\mathbf{d}_k^T \mathbf{h}_k}$

New gap $\mathbf{x}_{k+1} = \mathbf{x}_k + \alpha_k \mathbf{d}_k$

$\mathbf{mask_c} = \mathbf{x}_{k+1} > 0$

Admissible points $\mathbf{x}_{k+1}[\neg \mathbf{mask_c}] = 0$

if mean value **then**

$\mathbf{x}_{k+1} \leftarrow (\text{mean value}/mean(\mathbf{x}_{k+1}))\mathbf{x}_{k+1}$

end if

$\mathbf{r}_{k+1} \leftarrow \nabla f(\mathbf{x}_{k+1})$.

if mean value **then**

$\mathbf{mask_c} = x > 0$

$\mathbf{r}_{k+1} = \mathbf{r}_{k+1} - \overline{\mathbf{r}_{k+1}[\mathbf{mask_c}]}$

end if

$\mathbf{mask_{bound}} = \mathbf{x}_{k+1}[\neg \mathbf{mask_c}]\ AND\ \mathbf{r}_{k+1} > 0.$

$\mathbf{r}_{k+1}[\mathbf{mask_{bound}}] = 0$

if mean value **then**

$\mathbf{mask\ nonzero} = \mathbf{r}_{k+1} \neq 0$

$\mathbf{r}_{k+1}[\mathbf{mask\ nonzero}] = \mathbf{r}_{k+1}[\mathbf{mask\ nonzero}] - \overline{\mathbf{r}_{k+1}[\mathbf{mask\ nonzero}]}$

end if

Compute $\beta_{k+1} = \dfrac{\mathbf{r}_{k+1}^T \mathbf{h}_k}{\mathbf{d}_k^T \mathbf{h}_k} \approx \dfrac{\mathbf{r}_{k+1}^T (\mathbf{r}_{k+1} - \mathbf{r}_k)}{\alpha_k \mathbf{d}_k \mathbf{h}_k}$

New descent direction $\mathbf{d}_{k+1}[\neg\mathbf{mask_{bound}}] = -\mathbf{r}_{k+1}[\neg\mathbf{mask_{bound}}] + \beta_{k+1}\mathbf{d}_k[\neg\mathbf{mask_{bound}}]$

New descent direction $\mathbf{d}_{k+1}[\mathbf{mask_{bound}}] = 0$

end while

Comparison between Constrained CG algorithms by Polonsky & Keer and Insa Lyon Group.

	Polonsky & Keer	Insa Lyon Group
Compute step length	$\mathbf{h}_k = HessProd(\mathbf{d}_k)$ If mean value then $\mathbf{h}_k = \mathbf{h}_k - \overline{\mathbf{h}_k[\mathbf{mask_c}]}$ $\alpha_k = -\frac{\mathbf{r}k[\mathbf{mask_c}]^T \mathbf{d}_k[\mathbf{mask_c}]}{\mathbf{h}_k[\mathbf{mask_c}]\mathbf{d}_k[\mathbf{mask_c}]}$	$\mathbf{h}_k = HessProd(\mathbf{d}_k)$ $\alpha_k = -\frac{\mathbf{r}_k^T.\mathbf{d}_k}{\mathbf{d}_k^T \mathbf{h}_k}$
Compute $\mathbf{x_{k+1}}$	$\mathbf{x}_{k+1}[\mathbf{mask_c}] = \mathbf{x}_k[\mathbf{mask_c}] + \alpha_k \mathbf{d}_k[\mathbf{mask_c}]$.	$\mathbf{x}_{k+1} = \mathbf{x}_k + \alpha_k \mathbf{d}_k$
Add points to constraint	$\mathbf{mask_c} = \mathbf{x}_{k+1} > 0$ $\mathbf{x}_{k+1}[\neg\mathbf{mask_c}] = 0$	$\mathbf{mask_c} = \mathbf{x}_{k+1} > 0$ $\mathbf{x}_{k+1}[\neg\mathbf{mask_c}] = 0$
Enforce mean value	—	$\mathbf{x}_{k+1} \leftarrow (\text{mean value}/mean(\mathbf{x}_{k+1}))\mathbf{x}_{k+1}$
Compute $\mathbf{r_{k+1}}$	—	$\mathbf{r}_{k+1} \leftarrow \nabla f(\mathbf{x}_{k+1})$
Enforce mean value	—	$\mathbf{r}_{k+1} \leftarrow \mathbf{r}_{k+1} - \overline{\mathbf{r}_{k+1}[\mathbf{mask_c}]}$
Removing points from constraint	$\mathbf{mask_{bound}} = \mathbf{x}_{k+1} == 0\ AND\ \mathbf{r}_k < 0$ If $\mathbf{mask_{bound}} == \emptyset$ then $\delta = 1$ else $\delta = 0$ and $\mathbf{x}_{k+1}[I_{ol}] \leftarrow \mathbf{x}_{k+1}[I_{ol}] - \alpha_k \mathbf{r}_k[I_{ol}]$	$\mathbf{mask_{bound}} = \mathbf{x}_{k+1} == 0\ AND\ \mathbf{r}_{k+1} > 0$ $\mathbf{r}_{k+1}[\mathbf{mask_{bound}}] = 0$
Enforce mean value	$\mathbf{x}_{k+1} \leftarrow (\text{mean value}/mean(\mathbf{x}_{k+1}))\mathbf{x}_{k+1}$	$\mathbf{mask_{nonzero}} = \mathbf{r}_{k+1} \neq 0$ $\mathbf{r}_{k+1}[\mathbf{mask_{nonzero}}] \leftarrow \mathbf{r}_{k+1}[\mathbf{mask_{nonzero}}] - \overline{\mathbf{r}_{k+1}[\mathbf{mask_{nonzero}}]}$
Compute $\mathbf{r_{k+1}}$	$\mathbf{r}_{k+1} \leftarrow \nabla f(\mathbf{x}_{k+1})$	—
Enforce mean value	$\mathbf{r}_{k+1} \leftarrow \mathbf{r}_{k+1} - \overline{\mathbf{r}_{k+1}[\mathbf{mask_c}]}$	—
Update search direction	$R_{k+1} = \mathbf{r}_{k+1}[\mathbf{mask_c}]^2$ $\beta_{k+1} = \delta(R_{k+1}/R_k)$ $\mathbf{d}_{k+1}[\mathbf{mask_c}] \leftarrow -\mathbf{r}_{k+1}[\mathbf{mask_c}] + \beta_{k+1}\mathbf{d}_k[\mathbf{mask_c}]$ $\mathbf{d}_{k+1}[\neg\mathbf{mask_c}] = 0$	$\beta_{k+1} = \frac{\mathbf{r}_{k+1}^T.\mathbf{h}_k}{\mathbf{d}_k^T.\mathbf{h}_k}$ $\mathbf{d}_{k+1}[\neg\mathbf{mask_{bound}}] = -\mathbf{r}_{k+1}[\neg\mathbf{mask_{bound}}] + \beta_{k+1}\mathbf{d}_k[\neg\mathbf{mask_{bound}}]$ $\mathbf{d}_{k+1}[\neg\mathbf{mask_{bound}}] = 0$

Compute step length = In the both the algorithms PK (3) and BUG (4) the step length α is computed using the same formula. The only difference is in PK (3), the $HessProd()$ is adjusted for the mean value constraint and only contact points are considered.

Compute $\mathbf{x_{k+1}}$ = The expression for updating the new step is again same in both the algorithms here apart from the fact that in PK(3) the computations are done only on the points in contact.

Enforce mean value = Applying the mean value constraint consists in removing the mean value in the residual and re scaling the variable. In BUG (4) the mean value on x is enforced directly after the update of $\mathbf{x}_k \rightarrow \mathbf{x}_{k+1}$ and updating $\mathbf{x}_{k+1}$ according to the contact points. While in PK (3) the mean value is enforced after updating $\mathbf{x}_{k+1}$ for contact region and overlapping points.

Add points to constraint = The constraint on x is applied according to the points in contact region i.e., $mask\ c = x > 0$ in PK(3). Only the points which fulfill $\mathbf{mask_c}$ are considered in the minimization process and the points that do not satisfy the constraint are set to zero.
While in BUG (4) same thing is done by directly considering the *not* $\mathbf{mask_c} = \mathbf{x} \leq 0$ and setting the points to 0 that satisfy the condition.

Removing points from constraint	= In PK this means that the new overlap region $\mathbf{mask_{bound}} = \mathbf{x}_{k+1} == 0\ AND\ \mathbf{r}_k < 0$ of non-contacting nodes is calculated and if it is None then algorithm continues to run as a conjugate gradient algorithm. While if overlap region $\mathbf{mask_{bound}}$ is NOT None then the algorithm runs as a steepest descent algorithm i.e., β is set to 0 and $\mathbf{x}_{k+1}$ is updated again. And in BUG the residual is updated according the non-contacting region where $\mathbf{mask_{bound}} = \mathbf{x}_{k+1} == 0\ AND\ \mathbf{r}_{k+1} > 0$ and then $\mathbf{r}_{k+1}[\mathbf{mask_{bound}}] = 0$.
Update search direction	= The parameter β for the update of the search direction is updated in different ways in the two algorithms. In Bug, the computation of β depends upon the $HessProd()$, residual and old descent direction. And is fixed to this values for every iteration. Which is not the case for PK. In PK β depends upon the ratio of the square of gradient on the points out of contact. and could be set to completely 0 if there are points in the two surfaces overlapping at non contacting nodes.

Algorithm 5 GFMD [5]

Initialise

Damping coefficient η for critical damping.

Time step $\Delta t = 0.001 * (\pi/\sqrt{\text{max stiffness}})$.

$\widetilde{\mathbf{u}_{old}} = 0$

while $\max abs\mathbf{force} < tol$ **do** :

$\widetilde{\mathbf{force}} \leftarrow -\nabla(\widetilde{\mathbf{u}})$

if $\mathbf{force}_{ext}$ **then**

$\widetilde{\mathbf{force}} = \widetilde{\mathbf{force}} + \widetilde{\mathbf{force}_{ext}}(\widetilde{\mathbf{u}})$

end if

$\widetilde{\mathbf{force}_{damping}} = \eta * (\widetilde{\mathbf{u}} - \widetilde{\mathbf{u}_{old}}) * (1/\Delta t)$

$\widetilde{\mathbf{force}} = \widetilde{\mathbf{force}} + \widetilde{\mathbf{force}_{damping}}$

$\widetilde{\mathbf{u}_{new}} = 2\widetilde{\mathbf{u}} - \widetilde{\mathbf{u}_{old}} + \widetilde{\mathbf{force}}\Delta t^2$

if $max(abs(\mathbf{force})) <= tol$ **then**

converged = True

end if

end while

Algorithm 6 FIRE [6]

Initialise

Velocity $v = v_{old} = 0$.

Minimum number of consecutive steps when $P > 0, N_{min} = 5$.

Factor for Δt *increment*, $f_{inc} = 1.1$ *and* Δt decrement, $f_{dec} = 0.5$.

Factor for α update, $f_\alpha = 0.99$ *and* $\alpha_{start} = 0.1$.

$\Delta t_{max} = 1000 * \Delta t$.

$\widetilde{\mathbf{force}} = -\nabla f(\tilde{\mathbf{x}})$

while $\widetilde{\mathbf{force}} < tol$ **do** :

$\widetilde{\mathbf{x}} \leftarrow \widetilde{\mathbf{x}} + \widetilde{\mathbf{v}_{old}}\Delta t + 0.5\widetilde{\mathbf{force}}\Delta t^2$

$\widetilde{\mathbf{force}_{new}} = -\nabla f(\widetilde{\mathbf{x}})$

if $\mathbf{force}_{ext}$ **then**
$\widetilde{\mathbf{force}} \leftarrow \widetilde{\mathbf{force}} + \widetilde{\mathbf{force}_{ext}}$
end if

$\widetilde{\mathbf{v}_{old}} \leftarrow \widetilde{v}$

$\widetilde{\mathbf{v}} \leftarrow \widetilde{\mathbf{v}} + 0.5(\widetilde{\mathbf{force}_{new}} + \widetilde{\mathbf{force}})\Delta t$

$\widetilde{\mathbf{force}} \leftarrow \widetilde{\mathbf{force}_{new}}$

$Power \leftarrow \widetilde{\mathbf{force}}.real \cdot \widetilde{\mathbf{v}}.real$

$\widetilde{\mathbf{v}} \leftarrow (1-\alpha)\widetilde{\mathbf{v}} + \alpha(\frac{\widetilde{\mathbf{force}}}{\|\widetilde{\mathbf{force}}\|})|\widetilde{\mathbf{v}}|$

if $Power > 0$ **then**

if $(time - time_{p-pos}) - \Delta t <= 1e^{-3}$ **then**

$steps_{positive} = steps_{positive} + 1$

end if

$time_{p-pos} = time$

if $steps_{positive} > N_{min}$ **then**

$\Delta t = min(\Delta t * f_{inc}, \Delta t_{max})$

$\alpha = \alpha * f_{\alpha}$

end if

end if

if $Power <= 0$ **then**

$\Delta t = \Delta t * f_{dec}$

$\mathbf{v} = 0$

$\alpha = \alpha_{start}$

$steps_{positive} = 0$

end if

if $max(abs(\mathbf{force}_{real})) <= tol$ **then**

converged = True

end if

end while

Algorithm 7 Mass Weighted FIRE-GFMD [3]

Initialise

Velocity $v = v_{old} = 0$.

Minimum number of consecutive steps when $P > 0, N_{min} = 5$.

Factor for Δt *increment*, $f_{inc} = 1.1$ *and* Δt decrement, $f_{dec} = 0.5$.

Factor for α update, $f_\alpha = 0.99$ *and* $\alpha_{start} = 0.1$.

$\Delta t_{max} = 1000 * \Delta t$.

$\widetilde{\mathbf{force}} = -\nabla f(\tilde{x})$

if mass weight **then**

$\widetilde{\mathbf{force}} \leftarrow \dfrac{\widetilde{\mathbf{force}}}{\text{mass weight}}$

end if

while $\max abs\widetilde{\mathbf{force}} < tol$ **do** :

$\widetilde{\mathbf{x}} \leftarrow \widetilde{\mathbf{x}} + \widetilde{\mathbf{v}_{old}}\Delta t + 0.5\widetilde{\mathbf{force}}\Delta t^2$

$\widetilde{\mathbf{force}_{new}} = -\nabla f(\widetilde{\mathbf{x}})$

if $\mathbf{force}_{ext}$ **then**

$\widetilde{\mathbf{force}} \leftarrow \widetilde{\mathbf{force}} + \widetilde{\mathbf{force}_{ext}}$

end if

if mass weight **then**

$\widetilde{\mathbf{force}} \leftarrow \dfrac{\widetilde{\mathbf{force}}}{\text{mass weight}}$

end if

$\widetilde{\mathbf{v}_{old}} \leftarrow \widetilde{\mathbf{v}}$

$\widetilde{\mathbf{v}} \leftarrow \widetilde{\mathbf{v}} + 0.5(\widetilde{\mathbf{force}_{new}} + \widetilde{\mathbf{force}})\Delta t$

$\widetilde{\mathbf{force}} \leftarrow \widetilde{\mathbf{force}_{new}}$

$Power \leftarrow \widetilde{\mathbf{force}}.real \cdot \widetilde{\mathbf{v}}.real$

$\widetilde{\mathbf{v}} \leftarrow (1-\alpha)\widetilde{\mathbf{v}} + \alpha(\dfrac{\widetilde{\mathbf{force}}}{\|\widetilde{\mathbf{force}}\|})|\widetilde{\mathbf{v}}|$

if $Power > 0$ **then**

if $(time - time_{p-pos}) - \Delta t <= 1e^{-3}$ **then**

$steps_{positive} = steps_{positive} + 1$

end if

$time_{p-pos} = time$

if $steps_{positive} > N_{min}$ **then**

$\Delta t = min(\Delta t * f_{inc}, \Delta t_{max})$

$\alpha = \alpha * f_{\alpha}$

end if

end if

if $Power <= 0$ **then**

$\Delta t = \Delta t * f_{dec}$

$v = 0$

$\alpha = \alpha_{start}$

$steps_{positive} = 0$

end if

if $max(abs(\mathbf{force}_{real})) <= tol$ **then**

converged = True

end if

end while