Asynchronous Non-Smooth Optimisation via Projective Splitting

Utkarsh Sharma and Kashish Goel
Indian Institute of Technology Delhi

Supervised by
Dr. Zev Woodstock
IOL Lab, Zuse Institute Berlin

Summer Internship Presentation
July 13, 2023

Outline

Introduction

In our project, we minimize a **sum of separable nonsmooth convex functions** with **nonsmooth convex coupling terms** which utilise a linear mixture of the components.

We utilise the **Projective Splitting Algorithm**[1] for solving our problem which uses the benefits of Asynchronous Programming, Iterative Block-Activation approach, nonsmooth functions and Splitting.

Our goal involves the implementation of the algorithm and investigation of its nature with different Hyperparameters, application to problems such as **Image Recovery** and **Group-Sparse Binary Classification**.

[1] Asynchronous Block-Iterative Primal-Dual Decomposition Methods for Monotone Inclusions - Patrick L. Combettes and Jonathan Eckstein, 2018

Definitions

Let $\mathcal{H}$ be a real Hilbert space with inner product $\langle \cdot \mid \cdot \rangle$ (e.g. $\mathcal{H} = \mathbb{R}^n$, with $\langle x \mid y \rangle = x^T y$).

Definitions

Let $\mathcal{H}$ be a real Hilbert space with inner product $\langle \cdot \mid \cdot \rangle$ (e.g. $\mathcal{H} = \mathbb{R}^n$, with $\langle x \mid y \rangle = x^T y$).

Let $\Gamma_0(\mathcal{H}) = \{ f : \mathcal{H} \to \mathbb{R} \cup \{+\infty\} \mid f$ is convex, lower-semicontinuous, and proper $\}$
(e.g. $f(x) = x^2$, ReLU, Hinge loss, $\|Ax + b\|$). [3]

Definitions

Let $\mathcal{H}$ be a real Hilbert space with inner product $\langle \cdot \mid \cdot \rangle$ (e.g. $\mathcal{H} = \mathbb{R}^n$, with $\langle x \mid y \rangle = x^T y$).

Let $\Gamma_0(\mathcal{H}) = \{ f : \mathcal{H} \to \mathbb{R} \cup \{+\infty\} \mid f$ is convex, lower-semicontinuous, and proper $\}$ (e.g. $f(x) = x^2$, ReLU, Hinge loss, $\|Ax + b\|$). [3]

The **direct sum**, sometimes called a product space, is denoted as $\bigoplus_{i=1}^{m} \mathcal{H}_i = \mathcal{H}_1 \times \ldots \times \mathcal{H}_m$, where $\mathcal{H}_i$ represents the i^{th} Hilbert space.

The inner product on the direct sum is defined as follows:

$$\langle (x_1, \ldots, x_m), (y_1, \ldots, y_m) \rangle = \sum_{i=1}^{m} \langle x_i, y_i \rangle_{\mathcal{H}_i}$$

where $(x_1, \ldots, x_m)$, $(y_1, \ldots, y_m) \in \bigoplus_{i=1}^{m} \mathcal{H}_i$.

The inner product on the direct sum is defined as follows:

$$\langle (x_1, \ldots, x_m), (y_1, \ldots, y_m) \rangle = \sum_{i=1}^{m} \langle x_i, y_i \rangle_{\mathcal{H}_i}$$

where $(x_1, \ldots, x_m)$, $(y_1, \ldots, y_m) \in \bigoplus_{i=1}^{m} \mathcal{H}_i$.

The **indicator function** of set C, denoted as $\iota_C : \mathcal{H} \to [-\infty, +\infty]$ is defined as:

$$\iota_C(x) = \begin{cases} 0 & \text{if } x \in C \\ +\infty & \text{if } x \notin C \end{cases} .$$

Proximal Operators are mathematical tools used in optimization algorithms to handle nonsmooth functions and incorporate penalties or constraints into the optimization process.

Proximal Operators are mathematical tools used in optimization algorithms to handle nonsmooth functions and incorporate penalties or constraints into the optimization process.

For a function $f : \mathbb{R}^n \to \mathbb{R} \cup \{+\infty\}$, the **proximal operator** of f with respect to a parameter $\lambda > 0$ is defined as:

Proximal Operators are mathematical tools used in optimization algorithms to handle nonsmooth functions and incorporate penalties or constraints into the optimization process.

For a function $f : \mathbb{R}^n \to \mathbb{R} \cup \{+\infty\}$, the **proximal operator** of f with respect to a parameter $\lambda > 0$ is defined as:

$$\text{prox}_{\lambda f}(x) = \arg\min_y \left(f(y) + \frac{1}{2\lambda} \|y - x\|^2 \right)$$

where $\|\cdot\|$ represents a suitable norm on $\mathbb{R}^n$.

Proximal Operators are mathematical tools used in optimization algorithms to handle nonsmooth functions and incorporate penalties or constraints into the optimization process.

For a function $f : \mathbb{R}^n \to \mathbb{R} \cup \{+\infty\}$, the **proximal operator** of f with respect to a parameter $\lambda > 0$ is defined as:

$$\text{prox}_{\lambda f}(x) = \arg\min_y \left(f(y) + \frac{1}{2\lambda} \|y - x\|^2 \right)$$

where $\|\cdot\|$ represents a suitable norm on $\mathbb{R}^n$.

Intuitively - proxes provide a way to minimise the function value of f over its domain, without staying too far from x (penalised by the norm term).

Proxes are especially useful because iteratively applying the Proximal Operator on x_n ensures that we reach a global minimizer *(Martinet, 1970)*.

This gives rise to iterative algorithms that converge to a global minimizer for functions in $\Gamma_0(\mathcal{H})$.

Specifically, we implement and analyse one such algorithm - Algorithm 4 mentioned in [2]. We mention the advantages of the specific algorithm later.

Problem Statement

$$\underset{(x_i)_{i\in I}\in\bigoplus \mathcal{H}_i}{\text{minimize:}} \quad \sum_{i\in I} f_i(x_i) + \sum_{k\in K} g_k\Big(\sum_{i\in I}(L_{ki}\cdot x_i)\Big) \qquad (\ddot\smile)$$

where $(\mathcal{H}_i)_{i\in I}$ and $(\mathcal{G}_k)_{k\in K}$ are real Hilbert spaces with
$I = \{1, .., m\}$ and $K = \{1, .., p\}$
$L_{ki} : \mathcal{H}_i \to \mathcal{G}_k$ are linear
$f_i \in \Gamma_0(\mathcal{H}_i)$, $g_k \in \Gamma_0(\mathcal{G}_k)$

This objective function above is minimised along with its dual problem.

Formulating some well known examples

- **Convex Feasibility Problem** can be stated as follows: Given non-empty and closed convex sets $C_1, C_2, \ldots, C_n$ in a vector space $\mathcal{H}$, find a point x such that:

$$x \in C_1 \cap C_2 \cap \ldots \cap C_n.$$

The objective is to determine if there exists a feasible solution, i.e., a point that satisfies all the constraints simultaneously.

Formulating some well known examples

- **Convex Feasibility Problem** can be stated as follows: Given non-empty and closed convex sets $C_1, C_2, \ldots, C_n$ in a vector space $\mathcal{H}$, find a point x such that:

$$x \in C_1 \cap C_2 \cap \ldots \cap C_n.$$

The objective is to determine if there exists a feasible solution, i.e., a point that satisfies all the constraints simultaneously. We can formulate this in terms of our problem (☺) as :

Formulating some well known examples

- **Convex Feasibility Problem** can be stated as follows: Given non-empty and closed convex sets $C_1, C_2, \ldots, C_n$ in a vector space $\mathcal{H}$, find a point x such that:

$$x \in C_1 \cap C_2 \cap \ldots \cap C_n.$$

The objective is to determine if there exists a feasible solution, i.e., a point that satisfies all the constraints simultaneously.
We can formulate this in terms of our problem (☺) as :
$m = 1$
$f_1 = \mathbf{1}_{C_1}$
$p = n - 1$
$g_k = \mathbf{1}_{C_{k+1}}$
$\forall k \in K, i \in I \;\; L_{ki} = \mathbf{I}d$ (Identity)
which gives us $\underset{x_1 \in \mathcal{H}}{\text{minimize}} \quad \sum_{i \in I} \iota_{C_i}(x_1)$

- **Lasso Problem**

$$\underset{\beta \in \mathbb{R}^n}{\text{minimize}} \ \frac{1}{2n} \|y - X\beta\|_2^2 + \lambda \|\beta\|_1 \ .$$

- **Linear SVM Training**

$$\underset{x \in \mathcal{H}}{\text{minimize}} \ \sum_{d \in \mathcal{D}_1} \max\{0, 1 - d \cdot |x|\} + \sum_{d \in \mathcal{D}_2} \max\{0, 1 + h \cdot |x|\} + \lambda \cdot \|x\|_1.$$

- **Spectroscopic Image Recovery** (spoiler alert) : As we will see later, we can construct a loss function for degraded images if we know the degradation operator and minimise it to recover images.
  These can all, also be formulated as our problem (☺).

Solution Methods

To solve our problem, we utilise the **Combettes-Eckstein algorithm** for finding a Kuhn-Tucker point.

Solution Methods

To solve our problem, we utilise the **Combettes-Eckstein algorithm** for finding a Kuhn-Tucker point.

The algorithm solves the problem without adding restrictions such as smoothness, differentiability or strong convexity.

Solution Methods

To solve our problem, we utilise the **Combettes-Eckstein algorithm** for finding a Kuhn-Tucker point.

The algorithm solves the problem without adding restrictions such as smoothness, differentiability or strong convexity.

The algorithm offers a number of advantages over pre-existing algorithms which are as follows.

Advantage over pre-existing algorithms

Our proximal algorithm offers the following features which can not be found simultaneously in the pre-existing algorithms:

1. **Splitting:** The algorithm allows us to find the minimizer without calculating the prox of $\sum_{i \in I} f_i(x_i) + \sum_{k \in K} g_k(y_k)$ when the individual prox for f_i and g_k are known using splitting, since the proximity operator of composite functions are typically not known.

Advantage over pre-existing algorithms

Our proximal algorithm offers the following features which can not be found simultaneously in the pre-existing algorithms:

1. **Splitting:** The algorithm allows us to find the minimizer without calculating the prox of $\sum_{i \in I} f_i(x_i) + \sum_{k \in K} g_k(y_k)$ when the individual prox for f_i and g_k are known using splitting, since the proximity operator of composite functions are typically not known.
2. **Block-Activation:** Using Block-Activation, we activate the prox of only a block of functions $f_1, \ldots f_m, g_1, \ldots g_p$ at each iteration. This is highly efficient in contrast to algorithms which activate all the functions in every single iteration.

Advantage over pre-existing algorithms

Our proximal algorithm offers the following features which can not be found simultaneously in the pre-existing algorithms:

1. **Splitting:** The algorithm allows us to find the minimizer without calculating the prox of $\sum_{i \in I} f_i(x_i) + \sum_{k \in K} g_k(y_k)$ when the individual prox for f_i and g_k are known using splitting, since the proximity operator of composite functions are typically not known.
2. **Block-Activation:** Using Block-Activation, we activate the prox of only a block of functions $f_1, \ldots f_m, g_1, \ldots g_p$ at each iteration. This is highly efficient in contrast to algorithms which activate all the functions in every single iteration.
3. **Asynchrony:** The algorithm is asynchronous i.e., it is implementable on parallel architectures. This enables us to move to the calculation of next set of prox on a core without waiting for the other cores to finish the calculation.

Algorithm Overview

The overall idea is to iteratively refine current half-space by generating points in the graphs of monotone operators (In this case our subdifferentials of functions) until it successfully encapsulates the Kuhn-Tucker solution set.
The coordination step of the method is to project the current iterate onto the recently constructed half-space.
The advantages of this approach are as follows:

1. Does not require $(\|L_{ki}\|)_{i,k}$.
2. Does not involve the inversion of linear operators.
3. No additional assumptions are imposed on the operators present.

Motivation

- The Combettes-Eckstein algorithm has been there for a long time but no one has ever implemented it in the full setting.
- Implementation was followed by testing of different strategies to pick the **hyperparameters** which has never been done before.

Motivation

- The Combettes-Eckstein algorithm has been there for a long time but no one has ever implemented it in the full setting.
- Implementation was followed by testing of different strategies to pick the **hyperparameters** which has never been done before.

Our goals are as follows :

- Implement the algorithm and benchmark the performance.
- Open Source the algorithm as there are *very* few pre-existing algorithms of this class.
- Compare the convergence behaviour of our algorithm with different hyperparameter settings and find out the best initial conditions.

Implementation

- The algorithm is implemented in **Julia**. The **dependencies** include Julia packages - `LinearAlgebra`, `Wavelets`, `Images`, `Random`, `FileIO`, `ImageView` and `ProximalOperators`.
- Features in our Implementation:
  - **Flexible block selection strategies** that can be input by the users. Some of the block selection strategies are as follows:
    1. **Cyclic Strategy**
    2. **Random with Pooling**
    3. **Greedy Block Selection** based on expensiveness of Proxes.
  - Linear Operator L_{ki} (as mentioned in (☺)), can be **input as a function** (along with its adjoint operator) instead of matrices. This allows faster computations and less storage space.
  - Works correctly for vector product spaces where **each Hilbert Space can be of different dimensions**.
  - **Flexible γ_n, μ_n strategies** that the user can choose from.
  - **Asynchronous** in nature.

- Time taken Asynchronously for independent tasks vs Sync.

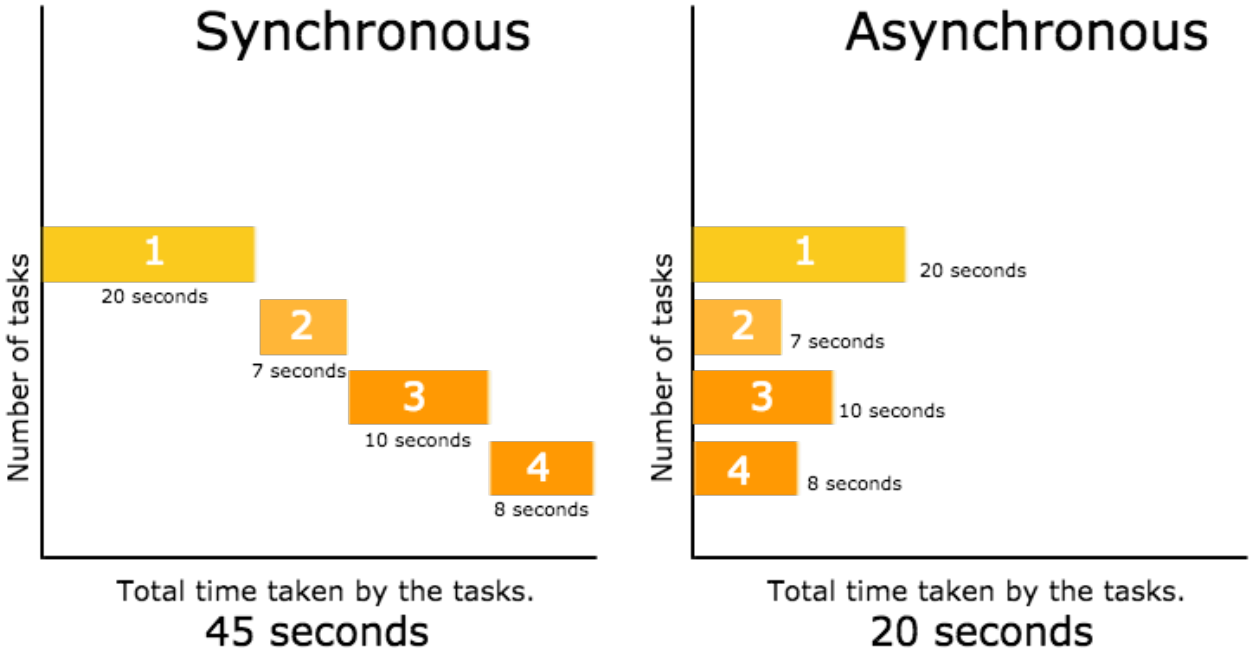
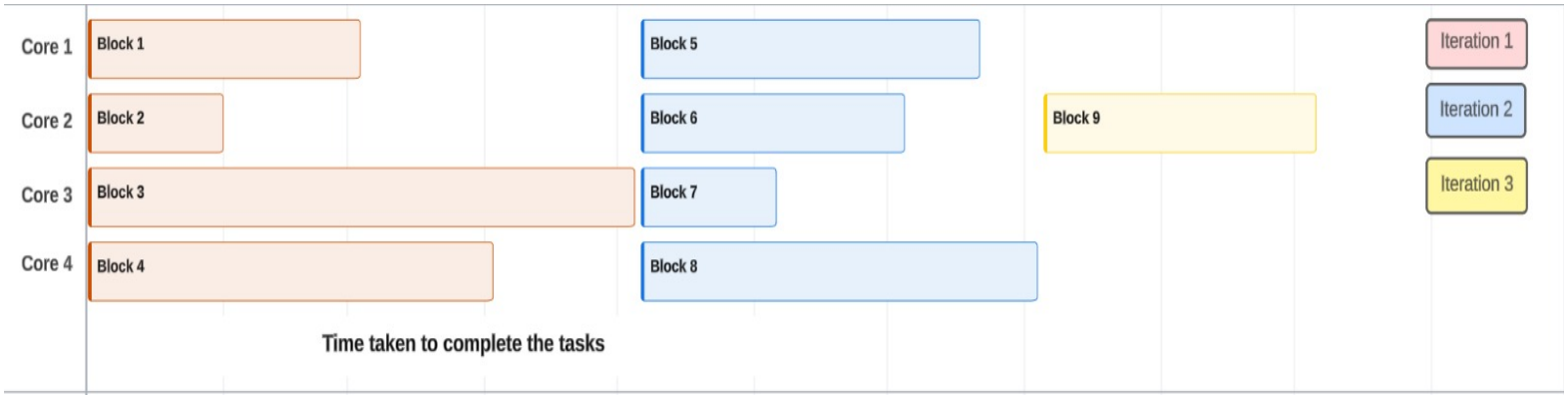
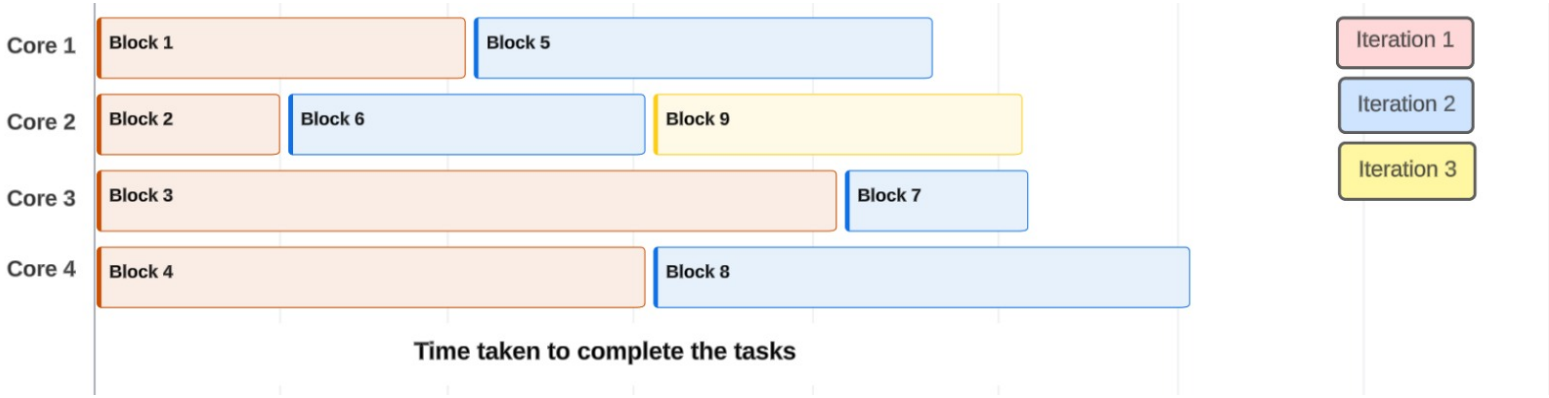


Diagram by Adrian Mejia

- The graphic below depicts the difference between block synchronous and block asynchronous algorithm



Block Synchronous Algorithm



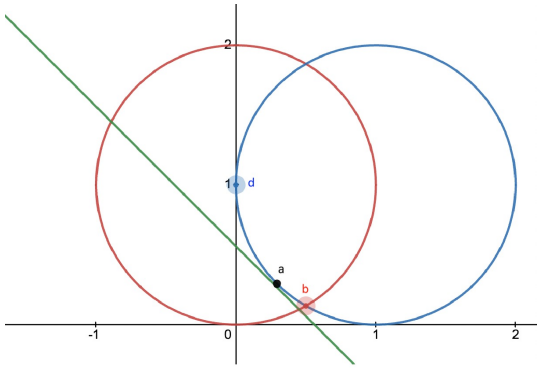
Block Asynchronous Algorithm

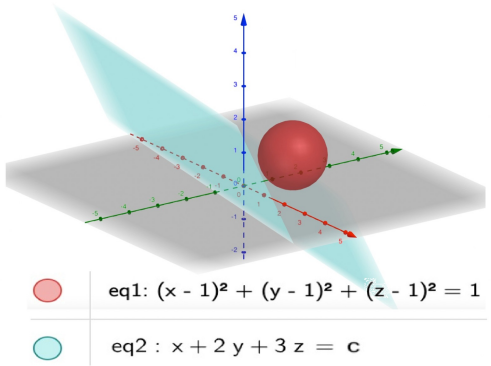
Examples Run

1. Let S_1, S_2 be a circles of radii 1 centered at $(0, 1)$ and $(1, 1)$ respectively in $\mathbb{R}^2$. Let S_3 be a sphere centered at $(1, 1, 1)$ in $\mathbb{R}^3$.
Problem:

$$\underset{\substack{x_1, x_2, x_3 \\ x_1, x_2 \in S_1 \cap S_2, x_3 \in S_3}}{\text{minimize}} \quad \langle [1, 0] \mid x_1 \rangle + \langle [0, 1] \mid x_2 \rangle + \langle [1, 2, 3] \mid x_3 \rangle + \iota_{(x_1 = x_2)}$$

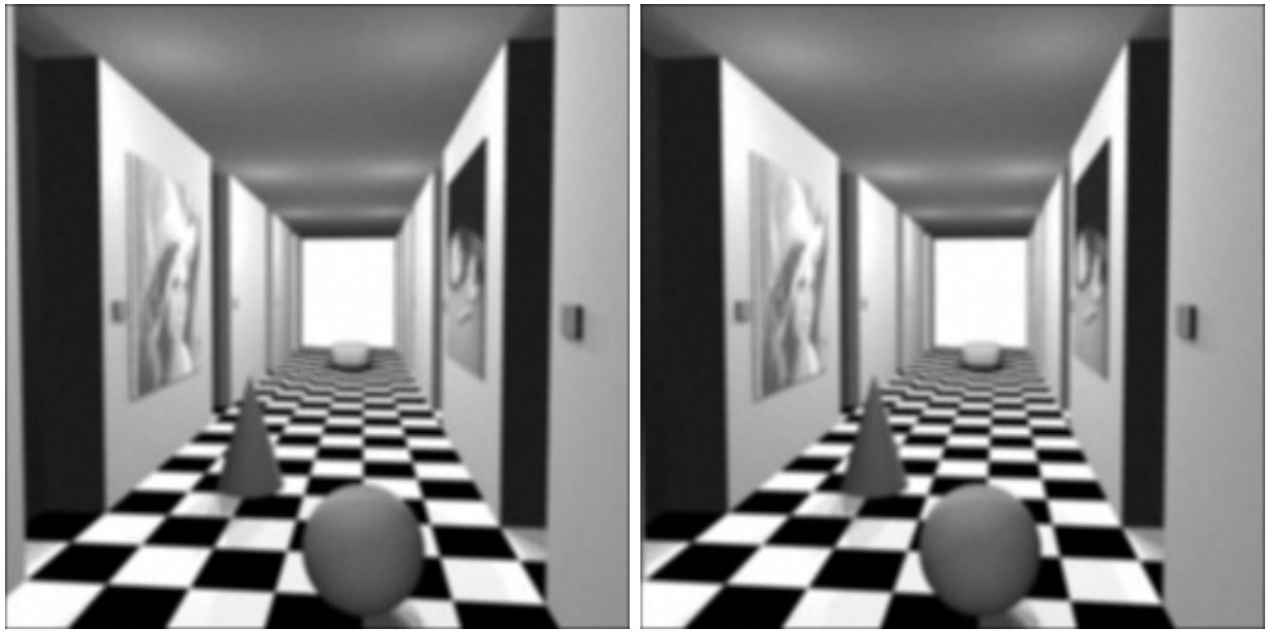
We get two slightly different problems depending on whether we include the indicator function term or not.





- Problem-1 is a combination of two problems (diagrams above) in different dimensions ($\mathbb{R}^2$ and $\mathbb{R}^3$) solved simultaneously.
- The answer is accurate within $\|x - x^*\| < 10^{-11}$ for both problems.
- True Values:
  1. Without $\iota_{(x_1=x_2)}$: $x_1 = [0, 1], x_2 = [\frac{1}{2}, 1 - \sqrt{\frac{3}{4}}]$, $x_3 = [1 - \frac{1}{\sqrt{14}}, 1 - \frac{2}{\sqrt{14}}, 1 - \frac{3}{\sqrt{14}}]$
  2. With $\iota_{(x_1=x_2)}$: $x_1 = x_2 = [1 - \sqrt{\frac{1}{2}}, 1 - \sqrt{\frac{1}{2}}]$, $x_3 = [1 - \frac{1}{\sqrt{14}}, 1 - \frac{2}{\sqrt{14}}, 1 - \frac{3}{\sqrt{14}}]$
- We reach the values within 10^{-11} error in < 1200 iterations

2. Consider the problem of restoring a pair of N-pixel Stereoscopic images $\overline{x_1}, \overline{x_2} \in \mathbb{R}^N$
The observations consist of degraded images
$z_1 = \mathcal{L}_1 \overline{x_1} + w_1$ and $z_2 = \mathcal{L}_2 \overline{x_2} + w_1$.
Where $\mathcal{L}_1$ and $\mathcal{L}_2$ are degradation operators (Gaussian Blurring in our case)



z_1 and z_2 ,i.e., the blurred images

We use the objective function mentioned in [1] to recover our images

$$\min_{x_1 \in \mathbb{R}^N, x_2 \in \mathbb{R}^N} \quad \sum_{k=1}^{N} \phi_{1,k}(\langle x_1 | e_{1,k} \rangle) + \sum_{k=1}^{N} \phi_{2,k}(\langle x_2 | e_{2,k} \rangle) + \frac{1}{2\sigma_1^2} \|\mathcal{L}_1 x_1 - z_1\|^2 + \frac{1}{2\sigma_2^2} \|\mathcal{L}_2 x_2 - z_2\|^2 + \frac{\vartheta}{2} \|x_1 - D x_2\|^2 \tag{1}$$

where $\phi_{i,k} = \mu_{i,k} |\cdot|$ for $i \in \{1, 2\}$ and $k \in [N]$

This can be formulated as (☺) by setting:

$f_1 : x_1 \longmapsto \langle (\mu_{1,k})_{k \in N} \,|\, \mathrm{DWT}(x_1)^2 \rangle$

$f_2 : x_2 \longmapsto \langle (\mu_{2,k})_{k \in N} \,|\, \mathrm{DWT}(x_2) \rangle$

$g_1 : y_1 \longmapsto \frac{1}{2\sigma_1^2} \|y_1 - z_1\|^2$

$g_2 : y_2 \longmapsto \frac{1}{2\sigma_2^2} \|y_2 - z_2\|^2$

$g_3 : y_3 \longmapsto \frac{\vartheta}{2} \|y_3\|^2$

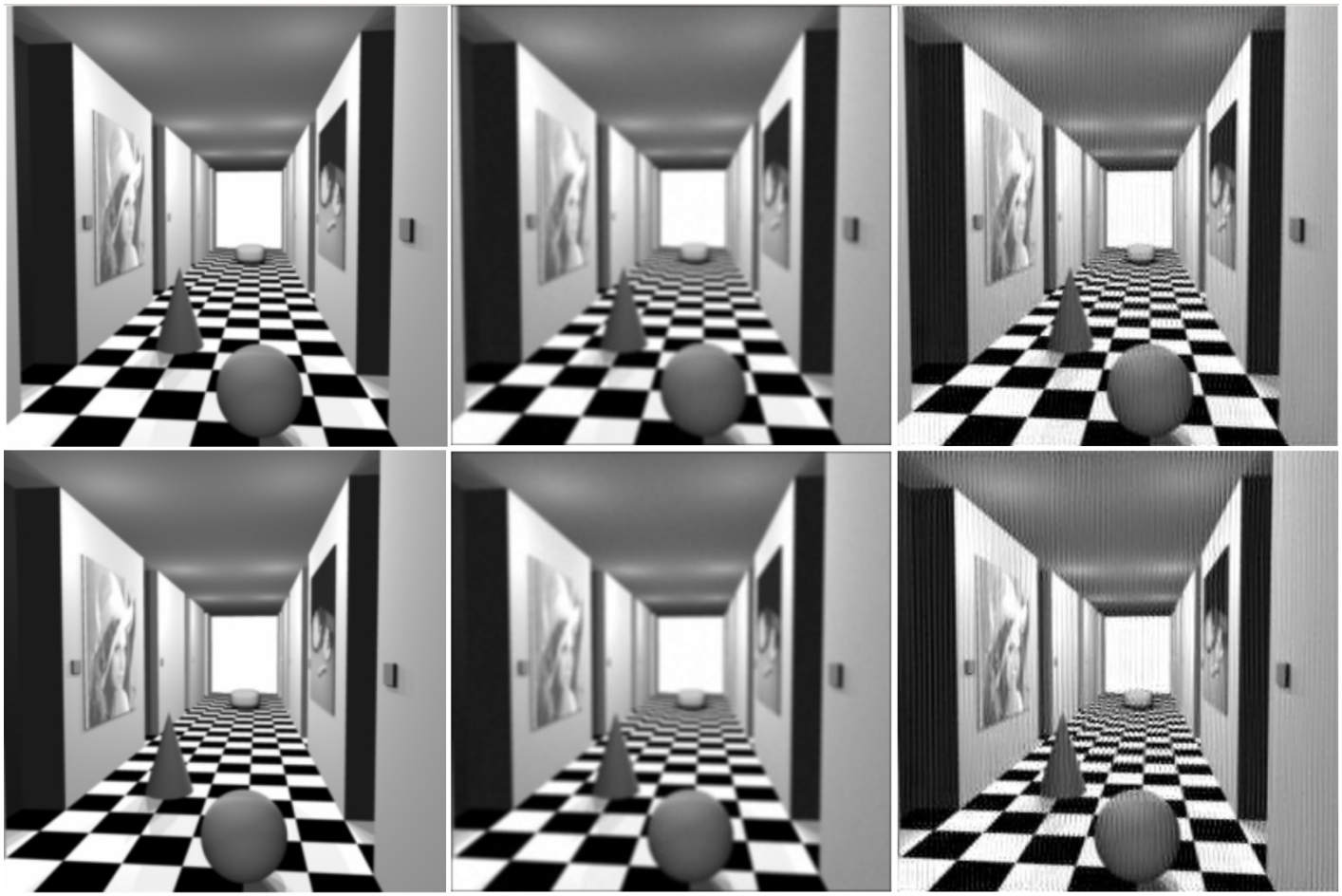
where $y_k = \sum_{i \in I} (L_{ki} \cdot x_i)$

where L_{ki} are functions : $H_i \to G_k$

$$\mathrm{L} = \begin{pmatrix} \mathcal{L}_1 & 0 \\ 0 & \mathcal{L}_2 \\ \mathbf{I_d} & -D \end{pmatrix}$$

[2] *DWT* refers to discrete wavelet transform in the symmlet bases with 4 vaishing moments

Results



Images in order (top to bottom and left to right) : Original Left and Right images , Degraded(Blurred) left and right , Recovered left and right

$\mu = 0.1$; $\theta = 0.1$; $\sigma_{noise} = 0.01$; $\sigma_{kernel} = 3.0$

Results

Images in order (top to bottom and left to right) : Original Left and Right images , Degraded(Blurred) left and right , Recovered left and right

$\mu = 0.1$; $\theta = 0.1$; $\sigma_{noise} = 0.01$; $\sigma_{kernel} = 4.0$

Results

The following 3x3 strategies were tested for the above example - 1

Hyperparameter Strategies			**Errors**[3]		
No.	**γ Strategy**	**μ strategy**	**x_1**	**x_2**	**x_3**
1	constant 1	constant 1	9.7e-15	1.4e-12	1.6e-11
2	constant 1	random constant	0.0	1.4e-12	1.6e-11
3	constant 1	decreasing seq.	2.2e-16	0.00065	0.0297
4	random constant	constant 1	0.0482	0.01236	1.1e-7
5	random constant	random constant	2.2e-14	1.4e-12	1.6e-11
6	random constant	decreasing seq.	0.0136	0.0018	0.0178
7	**decreasing seq.**	constant 1	2.2e-16	1.4e-12	1.6e-11
8	**decreasing seq.**	random constant	2.8e-14	1.4e-12	1.6e-11
9	**decreasing seq.**	decreasing seq.	4.1e-10	3.7e-7	2.0e-11

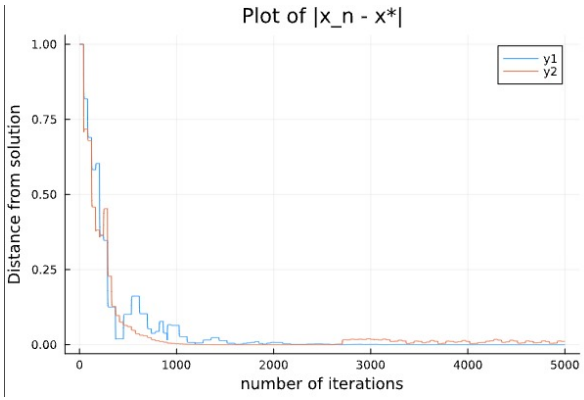
[3] Errors are the L2-Norm of the absolute difference between true solution and the solution obtained with the strategies given

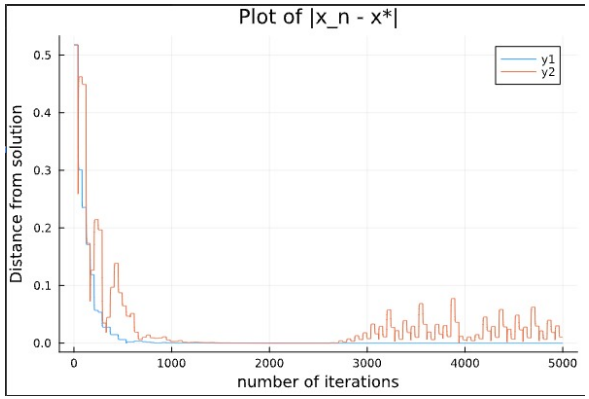
Results

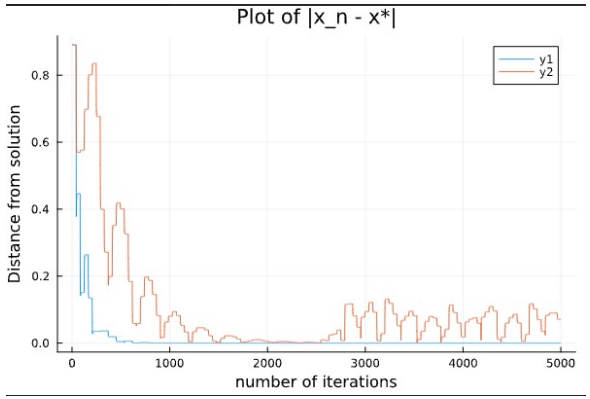
Note:

1. **Constant 1:** In this strategy, gamma or mu for all the functions equals 1
2. **Random Constant:** In this strategy, gamma or mu is a random constant in the range $[\epsilon, \frac{1}{\epsilon}]$ and is different for different functions
3. **Decreasing sequence:** In this strategy, gamma and mu form a decreasing sequence where the values start from $\frac{1}{\epsilon}$ and decrease with a common difference of 0.1 until it reaches the value, ϵ from where it becomes constant

Results







Plots for x_1, x_2, x_3 respectively where the distance from the minima is plotted at each iteration

y_1 : γ: Decreasing Sequence, μ: Constant 1

y_2 : γ: Random Constant, μ: Decreasing Sequence .

Thank you for your time!

References

L.M. Briceno-Arias, P.L. Combettes, J.-C. Pesquet, and N. Pustelnik.
Proximal algorithms for multicomponent image recovery problems.
Journal of Mathematical Imaging and Vision, 2011.

Jonathan Eckstein Patrick L. Combettes.
Asynchronous block-iterative primal-dual decomposition methods for monotone inclusions.
2018.

Zev Woodstock.
A crash course on proximity operators and nonsmooth convex optimization.
May, 2023.

Acknowledgements

- Our Advisor, Dr. Zev Woodstock
- Aryan Dua (Undergraduate student, IIT Delhi)