Explainable AI via Learning to Optimize

Howard Heaton[*†] **Samy Wu Fung**[*‡]

[†]Typal Research, Typal LLC
[‡]Department of Applied Mathematics and Statistics, Colorado School of Mines
research@typal.llc, swufung@mines.edu

Abstract

Indecipherable black boxes are common in machine learning (ML), but applications increasingly require explainable artificial intelligence (XAI). The core of XAI is to establish transparent and interpretable data-driven algorithms. This work provides concrete tools for XAI in situations where prior knowledge must be encoded and untrustworthy inferences flagged. We use the "learn to optimize" (L2O) methodology wherein each inference solves a data-driven optimization problem. Our L2O models are straightforward to implement, directly encode prior knowledge, and yield theoretical guarantees (*e.g.* satisfaction of constraints). We also propose use of interpretable certificates to verify whether model inferences are trustworthy. Numerical examples are provided in the applications of dictionary-based signal recovery, CT imaging, and arbitrage trading of cryptoassets.[1]

1. Introduction

A paradigm shift in machine learning is to construct explainable and transparent models, often called explainable AI (XAI)[66]. This is crucial for sensitive applications like medical imaging and finance (*e.g.* see recent work on the role of explainability[1;9;24;62]). Yet, many commonplace models (*e.g.* fully connected feed forward) offer limited interpretability. Prior XAI works give explanations via tools like sensitivity analysis[62] and layer-wise propagation,[10;50], but these neither quantify trustworthiness nor necessarily shed light on how to correct "bad" behaviours. Our work shows how learning to optimize (L2O) can be used to directly embed explainability into models.

The scope of this work is machine learning (ML) applications where domain experts can create approximate models by hand. In our setting, the inference $\mathcal{N}_\Theta(d)$ of a model $\mathcal{N}_\Theta$ with input d solves an optimization problem. That is, we use

$$\mathcal{N}_\Theta(d) \triangleq \underset{x\in\mathcal{C}_\Theta(d)}{\arg\min}\, f_\Theta(x;d), \tag{1}$$

where f_Θ is a function and $\mathcal{C}_\Theta(d) \subseteq \mathbb{R}^n$ is a constraint set (*e.g.* encoding prior information like physical quantities), and each (possibly) includes dependencies on weights Θ. Note the model $\mathcal{N}_\Theta$ is *implicit* since its output is defined by an optimality condition rather than an explicit computation.

[*]These authors contributed equally.

[1]Code is on Github: github.com/typal-research/xai-l2o

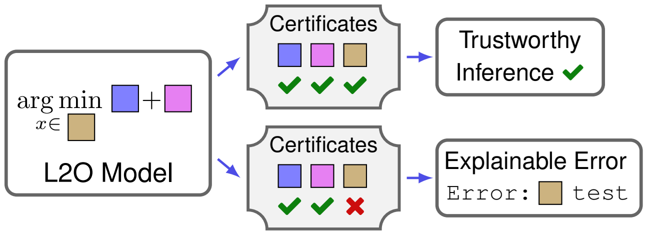

Figure 1: The L2O model is composed of parts (shown as colored blocks) that are analytic or learned. L2O inferences solve the optimization problem for given model inputs. Certificates label if each inference is consistent with training. If so, it is trustworthy; otherwise, the faulty model part errs.

A standard practice in software engineering is to code post-conditions after function calls return. Post-conditions are criteria used to validate what the user expects from the code and ensure code is not executed under the wrong assumptions.[4] We propose use of these for ML model inferences (see Figures 1 and 5). These conditions enable use of certificates with labels – pass, warning or fail – to describe each model inference. We define an inference to be *trustworthy provided it satisfies all provided post-conditions.*

Two ideas, optimization and certificates, form a concrete notion of XAI. Prior and data-driven knowledge can be encoded via optimization, and this encoding can be verified via certificates (see Figure 6). To illustrate, consider inquiring why a model generated a "bad" inference (*e.g.* an inference disagrees with observed measurements). The first diagnostic step is to check certificates. If no fails occurred, the model was not designed to handle the instance encountered. In this case, the model in (1) can be redesigned to encode prior knowledge of the situation. Alternatively, each failed certificate shows a type of error and often corresponds to portions of the model (see Figures 1 and 2). The L2O model allows debugging of algorithmic implementations and assumptions to correct errors. In a sense, this setup enables one to manually backpropagate errors to fix models (similar to training).

Contributions This work brings new explainability and guarantees to deep learning applications using prior knowledge. We propose novel implicit L2O models with intuitive design, memory efficient training, inferences that satisfy optimality/constraint conditions, and certificates that either indicate trustworthiness or flag inconsistent inference features.

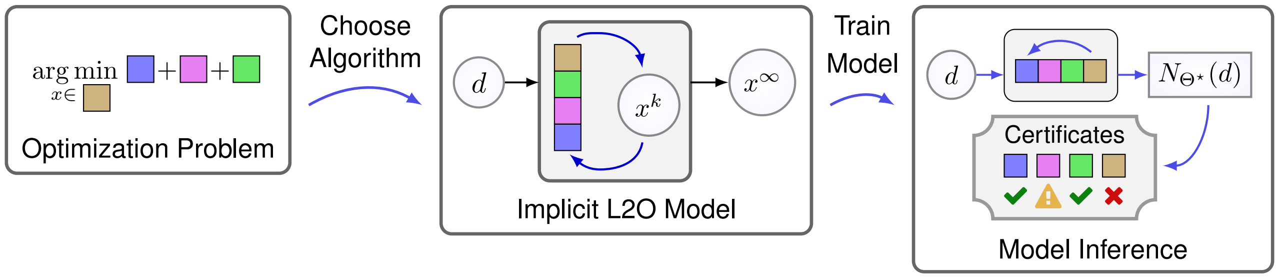

Figure 2: Left shows learning to optimize (L2O) model. Colored blocks denote prior knowledge and data-driven terms. Middle shows an iterative algorithm formed from the blocks (*e.g.* via proximal/gradient operators) to solve optimization problem. Right shows a trained model's inference $N_{\Theta^\star}(d)$ and its certificates. Certificates identify if properties of inferences are consistent with training data. Each label is associated with properties of specific blocks (indicated by labels next to blocks in right schematic). Labels take value pass ✓, warning ⚠, or fail ✖, and values identify if inference features for model parts are trustworthy.

L2O	Implicit	Flags	Model Property
✓			Intuitive Design
	✓		Memory Efficient
✓	✓		Satisfy Constraints + (above)
		✓	Trustworthy Inferences
✓	✓	✓	Explainable Errors + (above)

Table 1: Summary of design features and corresponding model properties. Design features yield additive properties, as indicated by "+ (above)." Proposed implicit L2O models with certificates have intuitive design, memory efficient training, inferences that satisfy optimality/constraint conditions, certificates of trustworthiness, and explainable errors.

Related Works

Closely related to our work is deep unrolling, a subset of L2O wherein models consist of a fixed number of iterations of a data-driven optimization algorithm. Deep unrolling has garnered great success and provides intuitive model design. We refer readers to recent surveys[3;18;49;64] for further L2O background. Downsides of unrolling are growing memory requirements with unrolling depth and a lack of guarantees.

Implicit models circumvent these two shortcomings by defining models using an equation (*e.g.* as in (1)) rather than prescribe a fixed number of computations as in deep unrolling. This enables inferences to be computed by iterating until convergence, thereby enabling theoretical guarantees. Memory-efficient training techniques were also developed for this class of models, which have been applied successfully in games,[34] music source separation,[41] language modeling,[11], segmentation,[12] and inverse problems.[30;32] The recent work[30] most closely aligns with our L2O methodology.

Related XAI works use labels/cards. Model Cards[48] document intended and appropriate uses of models. Care labels[51;52] are similar, testing properties like expressivity, runtime, and memory usage. FactSheets[8] are modeled after supplier declarations of conformity and aim to identify models' intended use, performance, safety, and security. These works provide statistics at the distribution level, complementing our work for trustworthiness of individual inferences.

2. Explainability via Optimization

Model Design The design of L2O models is naturally decomposed into two steps: optimization formulation and algorithm choice. The first step is to identify a tentative objective to encode prior knowledge via regularization (*e.g.* sparsity) or constraints (*e.g.* unit simplex for classification). We may also add terms that are entirely data-driven. Informally, this step identifies a special case of (1) of the form[2]

$$\mathcal{N}_\Theta(d) \triangleq \underset{x}{\arg\min}\ \text{(prior knowledge)} + \text{(data-driven terms)}. \tag{2}$$

The second design step is to choose an algorithm for solving the chosen optimization problem (*e.g.* proximal-gradient or ADMM[23]). We use iterative algorithms, and the update formula for each iteration is given by a *model operator* $T_\Theta(x; d)$. Updates are typically composed in terms of gradient and proximal operations. Some parameters (*e.g.* step sizes) may be included in the weights Θ to be tuned during training. Given data d, computation of the inference $\mathcal{N}_\Theta(d)$ is completed by generating a sequence $\{x_d^k\}$ via the relation

$$x_d^{k+1} = T_\Theta(x_d^k; d), \quad \text{for all } k \in \mathbb{N}. \tag{3}$$

By design, $\{x_d^k\}$ converges to a solution of (1), and we set[3]

$$\mathcal{N}_\Theta(d) = \lim_{k\to\infty} x_d^k. \tag{4}$$

In our context, each model inference $\mathcal{N}_\Theta(d)$ is defined to be an optimizer as in (1). Hence *properties of inferences can be interpreted via the optimization model* (1). The iterative algorithm is applied successively until stopping criteria are met (*i.e.* in practice we choose an iterate K, possibly dependent on d, so that $\mathcal{N}_\Theta(d) \approx x_d^K$). Because $\{x_d^k\}$ converges, we may adjust stopping criteria to approximate the limit to arbitrary precision, which implies we may provide guarantees on model inferences (*e.g.* satisfying a linear system of equations to a desired precision[30;32;34]). The properties of the implicit L2O model (1) are summarized by Table 1.

[2]Constraints are encoded in the objective using indicator functions, equaling 0 when constraint is satisfied and ∞ otherwise.

[3]If the optimization problem has multiple solutions, uniqueness of $\mathcal{N}_\Theta(d)$ is obtained in practice by fixing the initialization x_d^1.

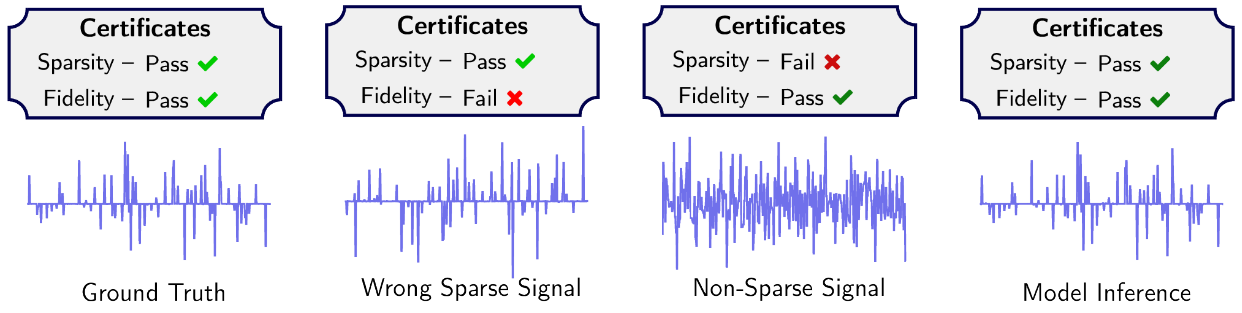

Figure 3: Example inferences for test data d. The sparsified version Kx of each inference x is shown (*c.f.* Figure 7) along with certificates. Ground truth was taken from test dataset of implicit dictionary experiment. The second from left is sparse and inconsistent with measurement data. The second from right complies with measurements but is not sparse. The rightmost is generated using our proposed model (IDM), which approximates the ground truth well and is trustworthy.

Example of Model Design. To make the model design procedure concrete, we illustrate this process on a classic problem: sparse recovery from linear measurements. These problems appear in many applications such as radar imaging[65] and speech recognition[28]. Here the task is to estimate a signal $x_d^\star$ via access to linear measurements d satisfying $d = Ax_d^\star$ for a known matrix A.

Step 1: Choose Model. Since true signals are known to be sparse, we include ℓ_1 regularization. To comply with measurements, we add a fidelity term. Lastly, to capture hidden features of the data distribution, we also add a data-driven regularization. Putting these together gives the problem

$$\min_{x\in\mathbb{R}^n} \underbrace{\tau\|x\|_1}_{\text{sparsity}} + \underbrace{\|Ax-d\|_2^2}_{\text{fidelity}} + \underbrace{\|W_1Ax\|^2 + \langle x, W_2 d\rangle}_{\text{data-driven regularizer}}, \quad (5)$$

where $\tau > 0$ and W_1 and W_2 are two tunable matrices. This model encodes a balance of three terms – sparsity, fidelity, data-driven regularization – each quantifiable via (5).

Step 2: Choose Algorithm. The proximal-gradient scheme generates a sequence $\{z^k\}$ converging to a limit which solves (5). By simplifying and combining terms, the proximal-gradient method can be written via the iteration[4]

$$z^{k+1} = \eta_{\tau\lambda}\big(z^k - \lambda W(Az^k - d)\big), \quad \text{for all } k \in \mathbb{N}, \quad (6)$$

where $\lambda > 0$ is a step-size and W is a matrix defined in terms[5] of W_1, W_2, and $A^\top$. From the update on the right hand side of (6), we see the step size λ can be "absorbed" into the tunable matrix W and the shrink function parameter can be set to $\theta > 0$. That is, this example model has weights $\Theta = (W, \theta, \tau)$ with model operator

$$T_\Theta(x; d) \triangleq \eta_\theta\big(x - W(Ax - d)\big), \quad (7)$$

which resembles the updates of previous L2O works.[19;31;46] Inferences are computed via a sequence $\{x_d^k\}$ with updates

$$x_d^{k+1} = T_\Theta(x_d^k; d), \quad \text{for all } k \in \mathbb{N}. \quad (8)$$

The model inference is the limit x_d^∞ of this sequence $\{x_d^k\}$.

[4]The shrink η_θ is given by $\eta_\theta(x) \triangleq \text{sign}(x)\max(|x| - \theta, 0)$.
[5]This assumes $W_2 = -A^\top W_1^\top W_1$.

Convergence Evaluation of the model $\mathcal{N}_\Theta(d)$ is well-defined and tractable under a simple assumption. By a classic result[42], it suffices to ensure, for all d, $T_\Theta(\cdot;\ d)$ is *averaged*, *i.e.* there is $\alpha \in (0,1)$ and Q such that $T_\Theta(x; d) = (1-\alpha)x + \alpha Q(x)$, where Q is 1-Lipschitz. When this property holds, the sequence $\{x_d^k\}$ in (3) converges to a solution $x_d^\star$. This may appear to be a strong assumption; however, common operations in convex optimization algorithms (*e.g.* proximals and gradient descent updates) are averaged. For entirely data-driven portions of T_Θ, several activation functions are 1-Lipschitz[20;27] (*e.g.* ReLU and softmax), and libraries like PyTorch[54] include functionality to force affine mappings to be 1-Lipschitz (*e.g.* spectral normalization). Furthermore, by making $T_\Theta(\cdot; d)$ a contraction, a unique fixed point is obtained. We emphasize, even without forcing T_Θ to be averaged, $\{x^k\}$ is often observed to converge in practice[11;30;32] upon tuning the weights Θ.

Trustworthiness Certificates

Explainable models justify whether each inference is trustworthy. We propose providing justification in the form of certificates, which verify various properties of the inference are consistent with those of the model inferences on training data and/or prior knowledge. Each certificate is a tuple of the form (name, label) with a property name and a corresponding label which has one of three values: pass, warning, or fail (see Figure 3). Each certificate label is generated by two steps. The first is to apply a function that maps inferences (or intermediate states) to a *nonnegative scalar value* α quantifying a property of interest. The second step is to map this scalar to a label. Labels are generated via the flow:

$$\text{Inference} \rightarrow \text{Property Value} \rightarrow \text{Certificate Label}. \quad (9)$$

Property Value Functions Several quantities may be used to generate certificates. In the model design example above, a sparsity property can be quantified by counting the number of nonzero entries in a signal, and a fidelity property can use the relative error $\|Ax - d\|/\|d\|$ (see Figure 3). To be most effective, property values are chosen to coincide with the optimization problem used to design the L2O model, *i.e.* to quantify structure of prior and data-driven knowledge. This

Concept	Quantity	Formula
Sparsity	Nonzeros	$\|x\|_0$
Measurements	Relative Error	$\|Ax - d\|/\|d\|$
Constraints	Distance to Set $\mathcal{C}$	$d_{\mathcal{C}}(x)$
Smooth Images	Total Variation	$\|\nabla x\|_1$
Classifier Confidence	Probability short of one-hot label	$1 - \max_i x_i$
Convergence	Iterate Residual	$\|x^k - x^{k-1}\|$
Regularization	Proximal Residual	$\|x - \text{prox}_{f_\Omega}(x)\|$

Table 2: Certificate examples. Each certificate is tied to a high-level concept, and then quantified in a formula. For classifier confidence, we assume x is in the unit simplex. The proximal is a data-driven update for f_Ω with weights Ω.

enables each certificate to clearly validate a portion of the model (see Figure 2). Since various concepts are useful for different types of modeling, we provide a brief (and non-comprehensive) list of concepts and possible corresponding property values in Table 2.

One property concept deserves particular attention: data-driven regularization. This regularization is important for discriminating between inference features that are qualitatively intuitive but difficult to quantify by hand. Rather than approximate a function, implicit L2O models directly approximate gradients/proximals. These provide a way to measure regularization indirectly via gradient norms/residual norms of proximals. Moreover, these norms (*e.g.* see last row of Table 2) are easy to compute and equal zero only at local minima of regularizers.[6] To our knowledge, this is the first work to *quantify* trustworthiness using the quality of inferences with respect to data-driven regularization.

Certificate Labels Typical certificate labels should follow a trend where inferences often obtain a pass label to indicate trustworthiness while warnings occur occasionally and failures are obtained in extreme situations. Let $\mathcal{A}$ be the sample space of model inference property values α generated from training and/or true data. We pick property value functions for which small α values are desirable and the distribution tail consists of larger α (see Figure 4). Intuitively, smaller property values of α resemble property values of inferences from training and/or test data. Thus, labels are assigned according to the probability of observing a value less than or equal to α, *i.e.* we evaluate the cumulative density function (CDF) defined for probability measure $\mathbb{P}_{\mathcal{A}}$ by

$$\text{CDF}(\alpha) = \int_0^\alpha \mathbb{P}_{\mathcal{A}}(\overline{\alpha})\ \mathrm{d}\overline{\alpha}, \tag{10}$$

Labels are chosen according to the task at hand. Let p_{p}, p_{w}, and $p_{\text{f}} = 1 - p_{\text{p}} - p_{\text{w}}$ be the probabilities for pass, warning,

[6]The learned proximal/gradient might not coincide with *any* function. However, this is of no concern since the operators still satisfy convergence properties and the model intuition still holds.

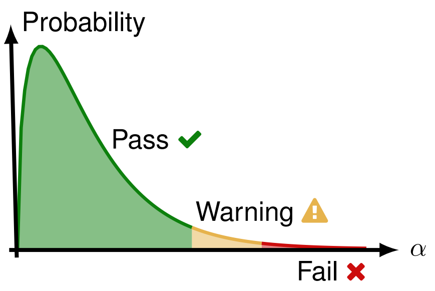

Figure 4: Example probability curve for property values α. Lower values of α are desired. Each value has a label: pass, warning, or fail. These are shown by green, yellow, and red.

Inference + Certificates → Trustworthy Inference

```
def TrustworthyInference(d):
    x, certs = model(d)
    if 'warning' in certs:
        warnings.warn('Warning Msg')
    if 'fail' in certs:
        raise Exception('Error Msg')
    return x
```

Figure 5: Example Python code to use certificates as post-conditions. Actual implementation should use specific warning/exception messages for flagged entries in `certs`.

and fail labels, respectively. Labels are made for α via

$$\text{Label}(\alpha) = \begin{cases} \text{pass} & \text{if CDF}(\alpha) < p_{\text{p}} \\ \text{warning} & \text{if CDF}(\alpha) \in [p_{\text{p}}, 1 - p_{\text{f}}) \\ \text{fail} & \text{otherwise.} \end{cases} \tag{11}$$

The remaining task is to estimate the CDF value for a given α. Recall we assume access is given to property values $\{\alpha_i\}_{i=1}^N$ from ground truths or inferences on training data, where N is the number of data points. To this end, given an α value, we may estimate its CDF value using

$$\text{CDF}(\alpha) \approx \frac{|\{\alpha_i : \alpha_i \leq \alpha,\ 1 \leq i \leq N\}|}{N} \tag{12a}$$

$$= \frac{\#\text{ of } \alpha_i\text{'s} \leq \alpha}{N}, \tag{12b}$$

where $|\cdot|$ denotes set cardinality. For large N, (12) well approximates the continuous CDF.

Certificate Implementation As noted in the introduction, trustworthiness certificates are evidence an inference satisfies post-conditions (*i.e.* passes various tests). Thus, they are to be used in code in the same manner as standard software engineering practice. Consider the snippet of code in Figure 5. As usual, an inference is generated by calling the model. However, alongside the inference `x`, certificates `certs` are returned that label whether the inference `x` passes tests that identify consistency with training data and prior knowledge.

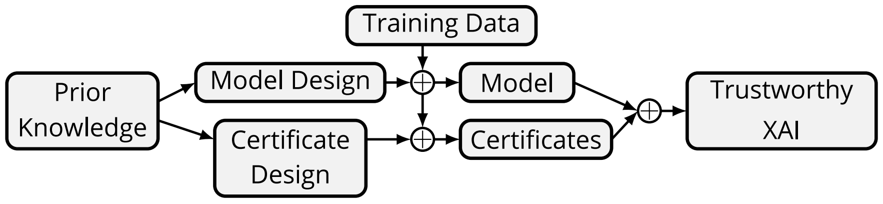

Figure 6: This diagram illustrates relationships between certificates, models, training data, and prior knowledge. Prior knowledge is embedded directly into model design via the L2O methodology. This also gives rise to quantities to measure for certificate design. The designed model is tuned using training data to obtain the "optimal" L2O model (shown by arrows touching top middle + sign). The certificates are tuned to match the test samples and/or model inferences on training data (shown by arrows with bottom middle + sign). Together the model and certificates yield inferences with certificates of trustworthiness.

3. Experiments

Each numerical experiment shows an application of novel implicit L2O models, which were designed directly from prior knowledge. Associated certificates of trustworthiness are used to emphasize the explainability of each model and illustrate use-cases of certificates. Experiments were coded using Python with the PyTorch library[54], the Adam optimizer[40], and, for ease of re-use, were run via Google Colab. We emphasize these experiments are for illustration of intuitive and novel model design and trustworthiness and are not benchmarked against state-of-the-art models.

Implicit Model Training

Standard backpropagation cannot be used for implicit models as it requires memory capacities beyond existing computing devices. Indeed, storing gradient data for each iteration in the forward propagation (see (3)) scales the memory during training linearly with respect to the number of iterations. Since the limit x^∞ solves a fixed point equation, implicit models can be trained by differentiating implicitly through the fixed point to obtain a gradient. This implicit differentiation requires further computations and coding. Instead of using gradients, we utilize Jacobian-Free Backpropagation (JFB)[26] to train models. JFB further simplifies training by only backpropagating through the final iteration, which was proven to yield preconditioned gradients. JFB trains using fixed memory (with respect to the K steps used to estimate $\mathcal{N}_\Theta(d)$) and avoids numerical issues arising from computing exact gradients,[13] making JFB and its variations[29;36] apt for training implicit models.

Implicit Dictionary Learning

Setup In practice, high dimensional signals often approximately admit low dimensional representations[16;43;53;55;56;71]. For illustration, we consider a linear inverse problem where true data admit sparse representations. Here each signal $x_d^\star \in \mathbb{R}^{250}$ admits a representation $s_d^\star \in \mathbb{R}^{50}$ via a transformation M (*i.e.* $x_d^\star = Ms_d^\star$). A matrix $A \in \mathbb{R}^{100\times 250}$ is applied to each signal $x_d^\star$ to provide linear measurements $d = Ax_d^\star$. Our task is to recover $x_d^\star$ given knowledge of A and d *without* the matrix M. Since the linear system is quite under-determined, schemes solely minimizing measurement error (*e.g.* least squares approaches) fail to recover true signals; additional knowledge is essential.

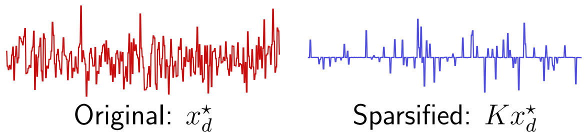

Figure 7: Training IDM yields sparse representation of inferences. Diagram shows a sample true data x (left) from test dataset and its sparsified representation Kx (right).

Model Design All convex regularization approaches are known lead to biased estimators whose expectation does not equal the true signal.[25] However, the seminal work[15] of Candes and Tao shows ℓ_1 minimization (rather than additive regularization) enables exact recovery under suitable assumptions. Thus, we minimize a sparsified signal subject to linear constraints via the implicit dictionary model (IDM)

$$\mathcal{N}_\Theta(d) \triangleq \operatorname*{arg\,min}_{x\in\mathbb{R}^{250}} \|Kx\|_1 \ \text{ s.t. } \ Ax = d. \qquad (13)$$

The square matrix K is used to leverage the fact x has a low-dimensional representation by transforming x into a sparse vector. Linearized ADMM[61] (L-ADMM) is used to create a sequence $\{x_d^k\}$ as in (3). The model $\mathcal{N}_\Theta$ has weights $\Theta = K$. If it exists, the matrix K^{-1} is known as a dictionary and $K\mathcal{N}_\Theta(d)$ is the corresponding sparse code; hence the name IDM for (13). To this end, we emphasize K is learned during training and is *different* from M, but these matrices are related since we aim for the product $Kx_d^\star = KMs_d^\star$ to be sparse. Note we use L-ADMM to *provably* solve (13), and $\mathcal{N}_\Theta$ is easy to train. More details are in Appendix B.

Discussion IDM combines intuition from dictionary learning with a reconstruction algorithm. Two properties are used to identify trustworthy inferences: sparsity and measurement compliance (*i.e.* fidelity). Sparsity and fidelity are quantified via the ℓ_1 norm of the sparsified inference (*i.e.* $K\mathcal{N}_\Theta(d)$) and relative measurement error. Figure 7 shows the training the model yields a sparsifying transformation K. Figure 3 shows the proposed certificates identify "bad" inferences that might, at first glance, appear to be "good" due to their compatibility with constraints. Lastly, observe the utility of learning K, rather than approximating M, is K makes it is easy to check if an inference admits a sparse representation. Using M to check for sparsity is nontrivial.

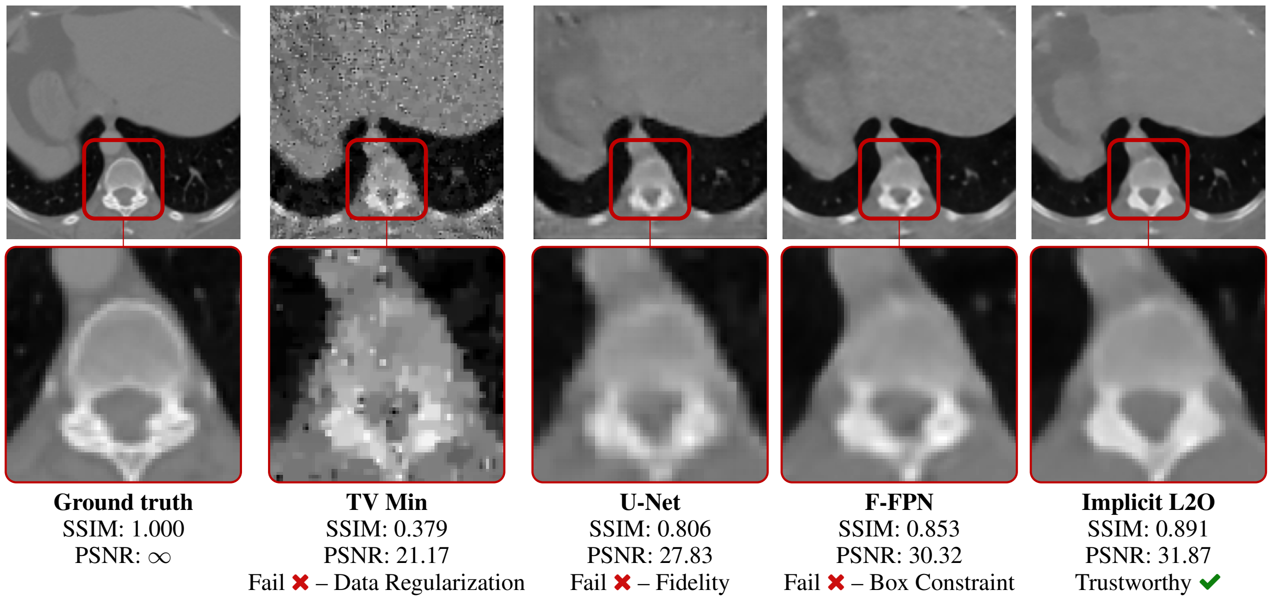

Figure 8: Reconstructions on test data computed via U-Net,[38] TV minimization, F-FPNs,[32] and Implicit L2O (left to right). Bottom row shows expansion of region indicated by red box. Pixel values outside $[0,1]$ are flagged. Fidelity is flagged when images do not comply with measurements, and regularization is flagged when texture features of images are sufficiently inconsistent with true data (*e.g.* grainy images). Labels are provided beneath each image (*n.b.* fail is assigned to images that are worse than 95% of L2O inferences on training data). Shown comparison methods fail while the Implicit L2O image passes all tests.

CT Image Reconstruction

Setup Comparisons are provided for low-dose CT examples derived from the Low-Dose Parallel Beam dataset (LoDoPab) dataset,[44] which consists of phantoms derived from actual human chest CT scans. Here CT measurements are simulated with a parallel beam geometry and a sparse-angle setup of only 30 angles and 183 projection beams, resulting in 5,490 equations and 16,384 unknowns. We add 1.5% Gaussian noise to *each individual beam measurement*. Images have resolution 128×128. To make errors easier to contrast between methods, the linear systems here are underdetermined and have more noise than those in some similar works. Image quality is determined using the Peak Signal-To-Noise Ratio (PSNR) and structural similarity index measure (SSIM). Mean squared error was used for the training loss. Training/test datasets consist of 20,000/2,000 samples.

Model Design The model for the CT experiment extends the IDM. In practice, it has been helpful to utilize a sparsifying transform.[37;69] We accomplish this via a linear operator K, which is applied and then this product is fed into a data-driven regularizer f_Ω with parameters Ω. We additionally ensure compliance with measurements from the Radon transform matrix A, up to a tolerance δ. In our setting, all pixel values are also known to be in the interval $[0,1]$. Combining our prior knowledge yields the implicit L2O model

$$\mathcal{N}_\Theta(d) \triangleq \operatorname*{arg\,min}_{x\in[0,1]^n} f_\Omega(Kx) \ \text{ s.t. } \ \|Ax-d\| \leq \delta \tag{14}$$

Here $\mathcal{N}_\Theta$ has weights $\Theta = (\Omega, K, \alpha, \beta, \lambda)$ with α, β and λ step-sizes in L-ADMM. More details are in Appendix C.

Discussion Comparisons of our method (Implicit L2O) with U-Net,[38] F-FPNs,[32] and total variation (TV) Minimization are given in Figure 8 and Table 3. Table 3 shows the average PSNR and SSIM reconstructions. Our model obtains the highest average PSNR and SSIM values on the test data while using 11% and 62% as many weights as U-Net and F-FFPN, indicating greater efficiency of the implicit L2O framework. Moreover, the L2O model is designed with three features: compliance with measurements (*i.e.* fidelity), valid pixel values, and data-driven regularization. Table 3 also shows the percentage of "fail" labels for these property values. Here, an inference fails if its property value is larger than 95% of the property values from the training/true data, *i.e.* we choose $p_p = 0.95$, $p_w = 0$, and $p_f = 0.05$ in (11). For the fidelity, our model never fails (due to incorporating the constraint into the network design). Our network fails 5.7% of the time for the data-driven regularization property. Overall, the L2O model generates the most trustworthy inferences. This is intuitive as this model outperforms the others and was specifically designed to embed all of our knowledge, unlike the others. To provide better intuition of the certificates, we also show the certificate labels for an image from the test dataset in Figure 8. The only image to pass all provided tests is the proposed implicit L2O model. This knowledge can help identify trustworthy inferences. Interestingly, the data-driven regularization enabled certificates to detect and flag "bad" TV Minimization features (*e.g.* visible staircasing effects[17;59]), which shows novelty of certificates as these features are intuitive, yet prior methods to quantify this were, to our knowledge, unknown.

Method	Avg. PSNR	Avg. SSIM	Box Constraint Fail	Fidelity Fail	Data Reg. Fail	# Params
U-Net	27.32 dB	0.761	5.75 %	96.95%	3.20%	533,593
TV Min	28.52 dB	0.765	0.00 %	0.00%	25.40%	4
F-FPN†	30.46 dB	0.832	47.15%	0.40%	5.05%	96,307
Implicit L2O	31.73 dB	0.858	0.00%	0.00%	5.70%	59,697

Table 3: Average PSNR/SSIM for CT reconstructions on the 2,000 image LoDoPab testing dataset. † Reported from original work.[32] U-Net was trained with filtered backprojection as in prior work.[38] Three properties are used to check trustworthiness: box constraints, compliance with measurements (*i.e.* fidelity), and data-driven regularization (via the proximal residual in Table 2). Failed sample percentages are numerically estimated via (12). Sample property values "fail" if they perform worse than 95% of the inferences on the training data, *i.e.* , its CDF value exceeds 0.95. Implicit L2O yields the most passes on test data.

Optimal Cryptoasset Trading

Setup Ethereum is a blockchain technology anyone can use to deploy permanent and immutable decentralized applications. This technology enables creation of decentralized finance (DeFi) primitives, which can give censorship-resistant participation in digital markets and expand the use of stable assets[39;57] and exchanges[35;67;70] beyond the realm of traditional finance. Popularity of cryptoasset trading (*e.g.* GRT and Ether) is exploding with the DeFi movement.[63;68]

Decentralized exchanges (DEXs) are a popular entity for exchanging cryptoassets (subject to a small transaction fee), where trades are conducted without the need for a trusted intermediary to facilitate the exchange. Popular examples of DEXs are constant function market makers (CFMMs),[6] which use mathematical formulas to govern trades. To ensure CFMMs maintain sufficient net assets, trades within CFMMs maintain constant total reserves (as defined by a function ϕ). A transaction in a CFMM tendering x assets in return for y assets with reserves assets r is accepted provided

$$\phi(r + \gamma x - y) \geq \phi(r), \tag{15}$$

with $\gamma \in (0, 1]$ a trade fee parameter. Here $r, x, y \in \mathbb{R}^n$ with each vector nonnegative and i-th entry giving an amount for the i-th cryptoasset type (*e.g.* Ether, GRT). Typical choices[5] of ϕ are weighted sums and products, *i.e.*

$$\phi(r) = \sum_{i=1}^{n} w_i r_i \ \text{ and } \ \phi(r) = \prod_{i=1}^{n} r_i^{w_i}. \tag{16}$$

where $w \in \mathbb{R}^n$ has positive entries.

This experiment aims to maximize arbitrage. Arbitrage is the simultaneous purchase and sale of equivalent assets in multiple markets to exploit price discrepancies between the markets. This can be a lucrative endeavor with cryptoassets.[47] For a given snapshot in time, our arbitrage goal is to identify a collection of trades that maximize the cryptoassets obtainable by trading between different exchanges, *i.e.* solve the (informal) optimization problem

$$\max_{\text{trade}} \text{Assets(trade)} \ \text{ s.t. } \ \text{trade} \in \{\text{valid trades}\}. \tag{17}$$

The set of valid trades is all trades satisfying the transaction rules for CFMMs given by (15) with nonnegative values for tokens tendered and received (*i.e.* $x, y \geq 0$). Prior works[5;7] deal with an idealistic noiseless setting while recognizing executing trades is not without risk (*e.g.* noisy information,

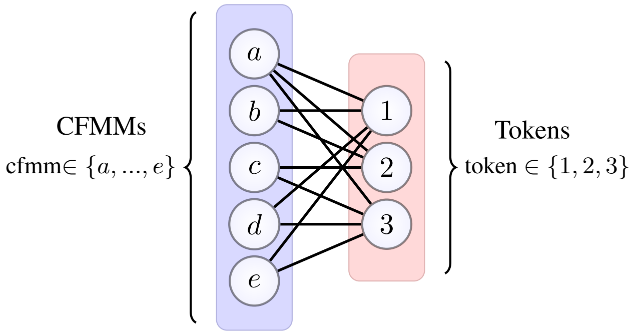

Figure 9: Network with 5 CFMMs and 3 tokens; structure replicates an experiment in recent work.[7] Black lines show available tokens for trade in each CFMM.

front running,[21] and trade delays). To show implications of trade risk, we incorporate noise in our trade simulations by adding noise $\varepsilon \in \mathbb{R}^n$ to CFMM asset observations, which yields noisy observed data $d = (1 + \varepsilon) \odot r$. Also, we consider trades with CFMMs where *several* assets can be traded simultaneously rather than restricting to pairwise swaps.

Model Design The aim is to create a model that infers a trade (x, y) maximizing utility. For a nonnegative vector $p \in \mathbb{R}^n$ of reference price valuations, this utility U is the net change in asset values provided by the trade, *i.e.*

$$U(x, y) \triangleq \underbrace{\sum_{j=1}^{m} \left\langle A^j p, A^j(y^j - x^j) \right\rangle}_{\text{net asset value change}}, \tag{18}$$

where A^j is a matrix mapping global coordinates of asset vector to the coordinates of the j-th CFMM (see Appendix D for details). For noisy data d, trade predictions can include a "cost of risk." This is quantified by regularizing the trade utility, *i.e.* introducing a penalty term. For matrices W^j, we model risk by a simple quadratic penalty via

$$U_\Theta(x, y) \triangleq U(x, y) - \underbrace{\frac{1}{2} \cdot \sum_{j=1}^{m} \|A^j W^j (x - y)\|^2}_{\text{risk model}}. \tag{19}$$

The implicit L2O model infers optimal trades via U_Θ, *i.e.*

$$\mathcal{N}_\Theta(d) \triangleq (x_d, y_d) = \operatorname*{arg\,max}_{(x,y) \in \mathcal{C}_\Theta(d)} U_\Theta(x, y), \tag{20}$$

Method	Predicted Utility	Executed Utility	Trade Execution	Risk Fail	Profitable Fail	# Params
Analytic	11.446	0.00	0.00%	100.00%	0%	0
Implicit L2O	0.665	0.6785	88.20%	3.6%	11.80%	126

Table 4: Averaged results on test data for trades in CFMM network. The analytic method always predicts a profitable trade, but fails to satisfy the constraints (due to noise). This failure is predicted by the certificates "risk" certificate and reflected by the 0% trade execution. Alternatively, the L2O scheme makes conservative predictions regarding constraints, which limits profitability. However, using these certificates, executed L2O trades are always profitable and satisfy constraints.

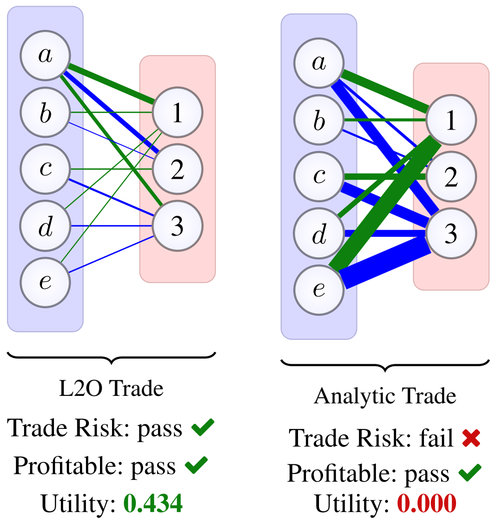

Figure 10: Example of proposed L2O (left) and analytic (right) trades with noisy data d. Blue and green lines show proposed cryptoassets x and y to tender and receive, respectively (widths show magnitude). The analytic trade is unable to account for trade risks, causing it to propose large trades that are *not* executed (giving executed utility of zero). This can be anticipated by the failed trade risk certificate. On the other hand, the L2O scheme is profitable (utility is 0.434) and is executed (consistent with the passing trade risk label).

where $\mathcal{C}_\Theta(d)$ encodes constraints for valid transactions. The essence of $\mathcal{N}_\Theta$ is to output solutions to (17) that account for transaction risks. A formulation of Davis-Yin operator splitting[22] is used for model evaluation. Further details of the optimization scheme are in Appendix D.

Discussion The L2O model contains three core features: profit, risk, and trade constraints. The model is designed to output trades that satisfy provided constraints, but note these are *noisy* and thus cannot be used to a priori determine whether a trade will be executed. For this reason, fail flags identify conditions to warn a trader when a trade should be aborted (due to an "invalid trade"). This can avoid wasting transaction fees (*i.e.* gas costs). Figure 10 shows an example of two trades, where we note the analytic method proposes a large trade that is *not* executed since it violates the trade constraints (due to noisy observations). The L2O method proposes a small trade that yielded arbitrage profits (*i.e.* $U > 0$) and has pass certificates. Comparisons are provided in Table 4 between the analytic and L2O models. Although the analytic method has "ideal" structure, it performs much worse than the L2O scheme. In particular, *no trades* are executable by the analytic scheme since the present noise always makes the proposed transactions fail to satisfy the actual CFMM constraints. Consistent wit this, every proposed trade by the analytic trade is flagged as risky in Table 4. The noise is on the order of 0.2% Gaussian noise of the asset totals.

4. Conclusions

Explainable ML models can be concretely developed by fusing certificates with the L2O methodology. The implicit L2O methodology enables prior and data-driven knowledge to be directly embedded into models, thereby providing clear and intuitive design. This approach is theoretically sound and compatible with state-of-the-art ML tools. The L2O model also enables construction of our certificate framework with easy-to-read labels, certifying if each inference is trustworthy. In particular, our certificates provide a principled scheme for the detection of inferences with "bad" features via data-driven regularization. Thanks to this optimization-based model design, failed certificates can be used to discard untrustworthy inferences and may help debugging the architecture. This reveals the interwoven nature of pairing implicit L2O with certificates. Our experiments illustrate these ideas in three different settings, presenting novel model designs and interpretable results.[7] Future work will study extensions to physics-based applications where PDE-based physics can be integrated into the model[45;58;60] as well as mechanism design applications in the blockchain space.[33]

Acknowledgments

The authors thank Wotao Yin, Stanley Osher, Daniel McKenzie, and Qiuwei Li for many fruitful discussions. Howard Heaton and Samy Wu Fung were supported by AFOSR MURI FA9550-18-1-0502 and ONR grants: N00014-18-1-2527, N00014-20-1-2093, and N00014-20-1-2787.

References

[1] Adadi, A.; and Berrada, M. 2018. Peeking inside the black-box: a survey on explainable artificial intelligence (XAI). *IEEE access*, 6: 52138–52160.

[2] Adler, J.; Kohr, H.; and Öktem, O. 2017. Operator Discretization Library (ODL).

[7] We emphasize each implicit L2O model presented is an example of an optimization model that can be used (*i.e.* not necessarily the "ideal" model).

[3] Amos, B. 2022. Tutorial on amortized optimization for learning to optimize over continuous domains. *arXiv preprint arXiv:2202.00665*.

[4] Anaya, M. 2018. *Clean Code in Python: Refactor your legacy code base*. Packt Publishing Ltd.

[5] Angeris, G.; Agrawal, A.; Evans, A.; Chitra, T.; and Boyd, S. 2021. Constant function market makers: Multi-asset trades via convex optimization. *arXiv preprint arXiv:2107.12484*.

[6] Angeris, G.; and Chitra, T. 2020. Improved price oracles: Constant function market makers. In *Proceedings of the 2nd ACM Conference on Advances in Financial Technologies*, 80–91.

[7] Angeris, G.; Chitra, T.; Evans, A.; and Boyd, S. 2021. Optimal Routing for Constant Function Market Makers.

[8] Arnold, M.; Bellamy, R. K.; Hind, M.; Houde, S.; Mehta, S.; Mojsilović, A.; Nair, R.; Ramamurthy, K. N.; Olteanu, A.; Piorkowski, D.; et al. 2019. FactSheets: Increasing trust in AI services through supplier's declarations of conformity. *IBM Journal of Research and Development*, 63(4/5): 6–1.

[9] Arrieta, A. B.; Díaz-Rodríguez, N.; Del Ser, J.; Bennetot, A.; Tabik, S.; Barbado, A.; García, S.; Gil-López, S.; Molina, D.; Benjamins, R.; et al. 2020. Explainable Artificial Intelligence (XAI): Concepts, taxonomies, opportunities and challenges toward responsible AI. *Information Fusion*, 58: 82–115.

[10] Bach, S.; Binder, A.; Montavon, G.; Klauschen, F.; Müller, K.-R.; and Samek, W. 2015. On pixel-wise explanations for non-linear classifier decisions by layer-wise relevance propagation. *PloS one*, 10(7): e0130140.

[11] Bai, S.; Kolter, J. Z.; and Koltun, V. 2019. Deep equilibrium models. *arXiv preprint arXiv:1909.01377*.

[12] Bai, S.; Koltun, V.; and Kolter, J. Z. 2020. Multiscale deep equilibrium models. *arXiv preprint arXiv:2006.08656*.

[13] Bai, S.; Koltun, V.; and Kolter, Z. 2021. Stabilizing Equilibrium Models by Jacobian Regularization. In Meila, M.; and Zhang, T., eds., *Proceedings of the 38th International Conference on Machine Learning*, volume 139 of *Proceedings of Machine Learning Research*, 554–565. PMLR.

[14] Beck, A. 2017. *First-Order Methods in Optimization*. SIAM.

[15] Candès, E. J.; Romberg, J.; and Tao, T. 2006. Robust uncertainty principles: Exact signal reconstruction from highly incomplete frequency information. *IEEE Transactions on information theory*, 52(2): 489–509.

[16] Carlsson, G.; Ishkhanov, T.; De Silva, V.; and Zomorodian, A. 2008. On the local behavior of spaces of natural images. *International journal of computer vision*, 76(1): 1–12.

[17] Chan, T.; Marquina, A.; and Mulet, P. 2000. High-order total variation-based image restoration. *SIAM Journal on Scientific Computing*, 22(2): 503–516.

[18] Chen, T.; Chen, X.; Chen, W.; Heaton, H.; Liu, J.; Wang, Z.; and Yin, W. 2021. Learning to optimize: A primer and a benchmark. *arXiv preprint arXiv:2103.12828*.

[19] Chen, X.; Liu, J.; Wang, Z.; and Yin, W. 2018. Theoretical linear convergence of unfolded ISTA and its practical weights and thresholds. *arXiv preprint arXiv:1808.10038*.

[20] Combettes, P. L.; and Pesquet, J.-C. 2020. Lipschitz certificates for layered network structures driven by averaged activation operators. *SIAM Journal on Mathematics of Data Science*, 2(2): 529–557.

[21] Daian, P.; Goldfeder, S.; Kell, T.; Li, Y.; Zhao, X.; Bentov, I.; Breidenbach, L.; and Juels, A. 2019. Flash boys 2.0: Frontrunning, transaction reordering, and consensus instability in decentralized exchanges. *arXiv preprint arXiv:1904.05234*.

[22] Davis, D.; and Yin, W. 2017. A three-operator splitting scheme and its optimization applications. *Set-valued and variational analysis*, 25(4): 829–858.

[23] Deng, W.; and Yin, W. 2016. On the global and linear convergence of the generalized alternating direction method of multipliers. *Journal of Scientific Computing*, 66(3): 889–916.

[24] Došilović, F. K.; Brčić, M.; and Hlupić, N. 2018. Explainable artificial intelligence: A survey. In *2018 41st International convention on information and communication technology, electronics and microelectronics (MIPRO)*, 0210–0215. IEEE.

[25] Fan, J.; and Li, R. 2001. Variable selection via nonconcave penalized likelihood and its oracle properties. *Journal of the American statistical Association*, 96(456): 1348–1360.

[26] Fung, S. W.; Heaton, H.; Li, Q.; McKenzie, D.; Osher, S.; and Yin, W. 2021. JFB: Jacobian-free backpropagation for implicit networks. *arXiv preprint arXiv:2103.12803*.

[27] Gao, B.; and Pavel, L. 2017. On the properties of the softmax function with application in game theory and reinforcement learning. *arXiv preprint arXiv:1704.00805*.

[28] Gemmeke, J. F.; Van Hamme, H.; Cranen, B.; and Boves, L. 2010. Compressive sensing for missing data imputation in noise robust speech recognition. *IEEE Journal of selected topics in Signal Processing*, 4(2): 272–287.

[29] Geng, Z.; Zhang, X.-Y.; Bai, S.; Wang, Y.; and Lin, Z. 2021. On Training Implicit Models. In *Thirty-Fifth Conference on Neural Information Processing Systems*.

[30] Gilton, D.; Ongie, G.; and Willett, R. 2021. Deep equilibrium architectures for inverse problems in imaging. *arXiv preprint arXiv:2102.07944*.

[31] Gregor, K.; and LeCun, Y. 2010. Learning fast approximations of sparse coding. In *Proceedings of the 27th international conference on international conference on machine learning*, 399–406.

[32] Heaton, H.; Fung, S. W.; Gibali, A.; and Yin, W. 2021. Feasibility-based fixed point networks. *arXiv preprint arXiv:2104.14090*.

[33] Heaton, H.; and Green, S. 2022. Equitable Continuous Organizations with Self-Assessed Valuations. *arXiv preprint arXiv:2203.10644*.

[34] Heaton, H.; McKenzie, D.; Li, Q.; Fung, S. W.; Osher, S.; and Yin, W. 2021. Learn to Predict Equilibria via Fixed Point Networks. *arXiv preprint arXiv:2106.00906*.

[35] Hertzog, E.; Benartzi, G.; and Benartzi, G. 2017. Bancor protocol. *White Paper*. storage.googleapis.com/website-bancor/2018/04/01ba8253-bancor_protocol_whitepaper_en.pdf (accessed on April 24 2022).

[36] Huang, Z.; Bai, S.; and Kolter, J. Z. 2021. Implicit^2: Implicit Layers for Implicit Representations. *Advances in Neural Information Processing Systems*, 34.

[37] Jiang, C.; Zhang, Q.; Fan, R.; and Hu, Z. 2018. Super-resolution ct image reconstruction based on dictionary learning and sparse representation. *Scientific reports*, 8(1): 1–10.

[38] Jin, K. H.; McCann, M. T.; Froustey, E.; and Unser, M. 2017. Deep convolutional neural network for inverse problems in imaging. *IEEE Transactions on Image Processing*, 26(9): 4509–4522.

[39] Kamvar, S.; Olszewski, M.; and Reinsberg, R. 2019. Celo: A multi-asset cryptographic protocol for decentralized social payments. *White Paper*. storage. googleapis. com/celo whitepapers/Celo A Multi Asset Cryptographic Protocol for Decentralized Social Payments.pdf.

[40] Kingma, D. P.; and Ba, J. 2014. Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980*.

[41] Koyama, Y.; Murata, N.; Uhlich, S.; Fabbro, G.; Takahashi, S.; and Mitsufuji, Y. 2021. Music Source Separation with Deep Equilibrium Models. *arXiv preprint arXiv:2110.06494*.

[42] Krasnosel'skii, M. 1955. Two remarks about the method of successive approximations. *Uspekhi Mat. Nauk*, 10: 123–127.

[43] Lee, A. B.; Pedersen, K. S.; and Mumford, D. 2003. The nonlinear statistics of high-contrast patches in natural images. *International Journal of Computer Vision*, 54(1-3): 83–103.

[44] Leuschner, J.; Schmidt, M.; Baguer, D. O.; and Maaß, P. 2019. The LoDoPaB-CT dataset: A benchmark dataset for low-dose CT reconstruction methods. *arXiv preprint arXiv:1910.01113*.

[45] Lin, A. T.; Fung, S. W.; Li, W.; Nurbekyan, L.; and Osher, S. J. 2021. Alternating the population and control neural networks to solve high-dimensional stochastic mean-field games. *Proceedings of the National Academy of Sciences*, 118(31).

[46] Liu, J.; and Chen, X. 2019. ALISTA: Analytic weights are as good as learned weights in LISTA. In *International Conference on Learning Representations (ICLR)*.

[47] Makarov, I.; and Schoar, A. 2020. Trading and arbitrage in cryptocurrency markets. *Journal of Financial Economics*, 135(2): 293–319.

[48] Mitchell, M.; Wu, S.; Zaldivar, A.; Barnes, P.; Vasserman, L.; Hutchinson, B.; Spitzer, E.; Raji, I. D.; and Gebru, T. 2019. Model cards for model reporting. In *Proceedings of the conference on fairness, accountability, and transparency*, 220–229.

[49] Monga, V.; Li, Y.; and Eldar, Y. C. 2021. Algorithm unrolling: Interpretable, efficient deep learning for signal and image processing. *IEEE Signal Processing Magazine*, 38(2): 18–44.

[50] Montavon, G.; Binder, A.; Lapuschkin, S.; Samek, W.; and Müller, K.-R. 2019. Layer-wise relevance propagation: an overview. *Explainable AI: interpreting, explaining and visualizing deep learning*, 193–209.

[51] Morik, K.; Kotthaus, H.; Heppe, L.; Heinrich, D.; Fischer, R.; Mücke, S.; Pauly, A.; Jakobs, M.; and Piatkowski, N. 2021. Yes We Care!–Certification for Machine Learning Methods through the Care Label Framework. *arXiv preprint arXiv:2105.10197*.

[52] Morik, K.; Kotthaus, H.; Heppe, L.; Heinrich, D.; Fischer, R.; Pauly, A.; and Piatkowski, N. 2021. The Care Label Concept: A Certification Suite for Trustworthy and Resource-Aware Machine Learning. *arXiv preprint arXiv:2106.00512*.

[53] Osher, S.; Shi, Z.; and Zhu, W. 2017. Low dimensional manifold model for image processing. *SIAM Journal on Imaging Sciences*, 10(4): 1669–1690.

[54] Paszke, A.; Gross, S.; Massa, F.; Lerer, A.; Bradbury, J.; Chanan, G.; Killeen, T.; Lin, Z.; Gimelshein, N.; Antiga, L.; et al. 2019. Pytorch: An imperative style, high-performance deep learning library. *Advances in neural information processing systems*, 32: 8026–8037.

[55] Peyré, G. 2008. Image processing with nonlocal spectral bases. *Multiscale Modeling & Simulation*, 7(2): 703–730.

[56] Peyré, G. 2009. Manifold models for signals and images. *Computer vision and image understanding*, 113(2): 249–260.

[57] Project, M. 2020. The Maker Protocol: MakerDAO's Multi-Collateral Dai (MCD) System. *White Paper*. storage. googleapis. com/celo whitepapers/Celo A Multi Asset Cryptographic Protocol for Decentralized Social Payments.pdf.

[58] Raissi, M.; Perdikaris, P.; and Karniadakis, G. E. 2019. Physics-informed neural networks: A deep learning framework for solving forward and inverse problems involving nonlinear partial differential equations. *Journal of Computational Physics*, 378: 686–707.

[59] Ring, W. 2000. Structural properties of solutions to total variation regularization problems. *ESAIM: Mathematical Modelling and Numerical Analysis*, 34(4): 799–810.

[60] Ruthotto, L.; Osher, S. J.; Li, W.; Nurbekyan, L.; and Fung, S. W. 2020. A machine learning framework for solving high-dimensional mean field game and mean field control problems. *Proceedings of the National Academy of Sciences*, 117(17): 9183–9193.

[61] Ryu, E.; and Yin, W. 2022. *Large-Scale Convex Optimization: Algorithm Designs via Monotone Operators*. Cambridge University Press.

[62] Samek, W.; and Müller, K.-R. 2019. Towards explainable artificial intelligence. In *Explainable AI: interpreting, explaining and visualizing deep learning*, 5–22. Springer.

[63] Schär, F. 2021. Decentralized finance: On blockchain- and smart contract-based financial markets. *FRB of St. Louis Review*.

[64] Shlezinger, N.; Whang, J.; Eldar, Y. C.; and Dimakis, A. G. 2020. Model-based deep learning. *arXiv preprint arXiv:2012.08405*.

[65] Siddamal, K.; Bhat, S. P.; and Saroja, V. 2015. A survey on compressive sensing. In *2015 2nd International Conference on Electronics and Communication Systems (ICECS)*, 639–643. IEEE.

[66] Van Lent, M.; Fisher, W.; and Mancuso, M. 2004. An explainable artificial intelligence system for small-unit tactical behavior. In *Proceedings of the national conference on artificial intelligence*, 900–907. Menlo Park, CA; Cambridge, MA; London; AAAI Press; MIT Press; 1999.

[67] Warren, W.; and Bandeali, A. 2017. 0x: An open protocol for decentralized exchange on the Ethereum blockchain. *White Paper*. github.com/0xProject/whitepaper.

[68] Werner, S. M.; Perez, D.; Gudgeon, L.; Klages-Mundt, A.; Harz, D.; and Knottenbelt, W. J. 2021. Sok: Decentralized finance (defi). *arXiv preprint arXiv:2101.08778*.

[69] Xu, Q.; Yu, H.; Mou, X.; Zhang, L.; Hsieh, J.; and Wang, G. 2012. Low-dose X-ray CT reconstruction via dictionary learning. *IEEE transactions on medical imaging*, 31(9): 1682–1697.

[70] Zhang, Y.; Chen, X.; and Park, D. 2018. Formal specification of constant product (xy= k) market maker model and implementation. *White Paper*.

[71] Zhang, Z.; Xu, Y.; Yang, J.; Li, X.; and Zhang, D. 2015. A survey of sparse representation: algorithms and applications. *IEEE access*, 3: 490–530.

A Linearized ADMM Formulation

Two of the numerical examples utilize variations of ADMM. This section is dedicated to a derivation of linearized ADMM used to solve problems of the form

$$\min_{x\in\mathbb{R}^n} f(Kx) + h(x) \ \text{ s.t. } \ \|Mx - d\| \leq \delta, \tag{21}$$

where K and M are linear operators, $\delta > 0$ is a noise tolerance, and f and h are proximable. First observe (21) can be rewritten as

$$\min_{x,w} f(Kx) + h(x) + \delta_{B(d,\delta)}(w) \quad \text{s.t.} \quad Mx - w = 0. \tag{22}$$

Defining the concatenation $\xi = (p, w)$, the function

$$g(\xi) \triangleq f(p) + \delta_{B(d,\delta)}(w), \tag{23}$$

and $S = [K; M]$, yields

$$\min_{x,\xi} h(x) + g(\xi) \quad \text{s.t.} \quad Sx - \xi = 0. \tag{24}$$

Then linearized ADMM[61] yields

$$x^{k+1} = \text{prox}_{\beta h}\left(x^k - \beta S^\top(\nu^k + \alpha(Sx^k - \xi^k))\right) \tag{25a}$$
$$\xi^{k+1} = \text{prox}_{\lambda g}\left(\xi^k + \lambda(\nu^k + \alpha(Sx^{k+1} - \xi^k))\right) \tag{25b}$$
$$\nu^{k+1} = \nu^k + \alpha(Sx^{k+1} - \xi^{k+1}), \tag{25c}$$

where α, β, λ are step-sizes. Rearranging, we obtain

$$\xi^{k+1} = \text{prox}_{\lambda g}\left(\xi^k + \lambda(\nu^k + \alpha(Sx^k - \xi^k))\right) \tag{26a}$$
$$\nu^{k+1} = \nu^k + \alpha(Su^k - \xi^{k+1}) \tag{26b}$$
$$x^{k+1} = \text{prox}_{\beta h}\left(x^k - \beta S^\top(\nu^{k+1} + \alpha(Sx^k - \xi^{k+1}))\right). \tag{26c}$$

Expanding ξ^{k+1} reveals block-wise updates, *i.e.*

$$p^{k+1} = \text{prox}_{\lambda f}(p^k + \lambda(\nu_1^k + \alpha(Kx^k - p^k)) \tag{27a}$$
$$w^{k+1} = P_{B(d,\varepsilon)}(w^k + \lambda(\nu_2^k + \alpha(Mx^k - w^k))), \tag{27b}$$

where $\nu^k = (\nu_1^k, \nu_2^k)$ and $P_{B(d,\varepsilon_\theta)}$ is the projection onto the Euclidean ball of radius ε_θ centered about d. Writing out expanded forms gives

$$p^{k+1} = \text{prox}_{\lambda f}\left(p^k + \lambda(\nu_1^k + \alpha(Kx^k - p^k))\right) \tag{28a}$$
$$w^{k+1} = P_{B(d,\delta)}\left(w^k + \lambda(\nu_2^k + \alpha(Mx^k - w^k))\right) \tag{28b}$$
$$\nu_1^{k+1} = \nu_1^k + \alpha(Kx^k - p^{k+1}) \tag{28c}$$
$$\nu_2^{k+1} = \nu_2^k + \alpha(Mx^k - w^{k+1}) \tag{28d}$$
$$x^{k+1} = \text{prox}_{\beta h}\left(x^k - \beta S^\top(\nu^{k+1} + \alpha(Sx^k - \xi^{k+1}))\right) \tag{28e}$$

Expanding the update for x^k reveals

$$x^{k+1} = x^k - \beta S^\top(\nu^{k+1} + \alpha(Sx^k - \xi^{k+1})) \tag{29a}$$
$$= x^k - \beta \begin{bmatrix} K \\ M \end{bmatrix}^\top \begin{bmatrix} \nu_1^{k+1} + \alpha(Kx^k - p^{k+1}) \\ \nu_2^{k+1} + \alpha(Mx^k - w^{k+1}) \end{bmatrix} \tag{29b}$$
$$= x^k - \beta K^\top\left(\nu_1^{k+1} + \alpha(Kx^k - p^{k+1})\right) \tag{29c}$$
$$- \beta M^\top\left(\nu_2^{k+1} + \alpha(Mx^k - w^{k+1})\right) \tag{29d}$$
$$= x^k - \beta K^\top\left(2\nu_1^{k+1} - \nu_1^k\right) \tag{29e}$$
$$- \beta M^\top\left(2\nu_2^{k+1} - \nu_2^k\right). \tag{29f}$$

The final form we implement is the tuple of update relations

$$p^{k+1} = \text{prox}_{\lambda f}\left(p^k + \lambda(\nu_1^k + \alpha(Kx^k - p^k))\right) \tag{30a}$$

$$w^{k+1} = P_{B(d,\delta)}\left(w^k + \lambda(\nu_2^k + \alpha(Mx^k - w^k))\right) \tag{30b}$$

$$\nu_1^{k+1} = \nu_1^k + \alpha(Kx^k - p^{k+1}) \tag{30c}$$

$$\nu_2^{k+1} = \nu_2^k + \alpha(Mx^k - w^{k+1}) \tag{30d}$$

$$r^k = K^\top\left(2\nu_1^{k+1} - \nu_1^k\right) + M^\top\left(2\nu_2^{k+1} - \nu_2^k\right) \tag{30e}$$

$$x^{k+1} = \text{prox}_{\beta h}\left(x^k - \beta r^k\right). \tag{30f}$$

B Supplement for Implicit Dictionary

Observe (13) is a special case of (21), taking $h = 0$, $M = A$, and $f = \|\cdot\|_1$. That is, in this case, we obtain the iteration

$$p^{k+1} = \eta_\lambda\left(p^k + \lambda(\nu_1^k + \alpha(Kx^k - p^k))\right) \tag{31a}$$

$$\nu_1^{k+1} = \nu_1^k + \alpha(Kx^k - p^{k+1}) \tag{31b}$$

$$\nu_2^{k+1} = \nu_2^k + \alpha(Ax^k - d) \tag{31c}$$

$$r^k = K^\top\left(2\nu_1^{k+1} - \nu_1^k\right) + A^\top\left(2\nu_2^{k+1} - \nu_2^k\right) \tag{31d}$$

$$x^{k+1} = x^k - \beta r^k, \tag{31e}$$

where η_λ is the shrink function.

C Supplement for CT Reconstruction

Observe (14) is a special case of (21), taking $h = \delta_{[0,1]^n}$, $M = A$, and $f = f_\Omega$. That is, in this case, we obtain the iteration

$$p^{k+1} = \text{prox}_{\lambda f_\Omega}\left(p^k + \lambda(\nu_1^k + \alpha(Kx^k - p^k))\right) \tag{32a}$$

$$w^{k+1} = P_{B(d,\delta)}\left(w^k + \lambda(\nu_2^k + \alpha(Ax^k - w^k))\right) \tag{32b}$$

$$\nu_1^{k+1} = \nu_1^k + \alpha(Kx^k - p^{k+1}) \tag{32c}$$

$$\nu_2^{k+1} = \nu_2^k + \alpha(Ax^k - w^{k+1}) \tag{32d}$$

$$r^k = K^\top\left(2\nu_1^{k+1} - \nu_1^k\right) + A^\top\left(2\nu_2^{k+1} - \nu_2^k\right) \tag{32e}$$

$$x^{k+1} = P_{[0,1]^n}\left(x^k - \beta r^k\right). \tag{32f}$$

TV minimization is obtained from (14) by letting f_Ω be the ℓ_1 norm and K be a discrete differencing operator. For comparison to an analytic method, we use anisotropic TV minimization, *i.e.*

$$\min_{u \in [0,1]^n} \|Du\|_1 \ \text{ s.t. } \|Au - d\| \leq \varepsilon, \tag{33}$$

where ε is hand-tuned. The Operator Discretization Library (ODL) Python library[2] is used to compute the filtered backprojections.

D Supplement for Cryptoasset Trades

This section is broken into three parts. The geometric constraint sets are of particular importance to handle in a decoupled fashion, and so the first subsection is dedicated to handling CFMM constraints for batches of transactions.[8] The second subsection then identifies the projection operations needed. This is followed by a derivation of a particular operator splitting used to solve the problem, giving explicit lists for updates.

[8]Note closed form expressions exist for pairwise swaps.

Constraint Formulation

The dimension of the vector space for each CFMM may differ since some exchanges might not provide access to particular cryptoassets. Consequently, we follow similarly to recent work[7] in using matrices $A^j \in \mathbb{R}^{n_j \times n}$ to convert global coordinates into the local coordinates of the j-th CFMM,[9] *i.e.*

$$A^j_{k\ell} \triangleq \begin{cases} 1 & \begin{array}{l}\text{if token } k \text{ is in the } j\text{-th CFMM's coordinates} \\ \text{is token } \ell \text{ is in global coordinates}\end{array} \\ 0 & \text{otherwise.} \end{cases} \tag{34}$$

Let $d \in \mathbb{R}^{n\times m}$ be a matrix with the j-th column d^j the reserve assets in the j-th CFMM. For weighted geometric CFMMs, set

$$\hat{d} \triangleq (1+\delta) \odot d \tag{35}$$

and

$$\alpha_j \triangleq \prod_{j=1}^{n_j} (\hat{d}_i^j)^{w_i^j}, \ \text{ for all } j \in [m], \tag{36}$$

where $\delta_j \geq 0$ is a tolerance, n_j is the number of asset types in the j-th CFMM, $w^j \in \mathbb{R}^{n_j}$ is a positive weighting, and

$$\mathcal{A}_j \triangleq \left\{ v \in \mathbb{R}^{n_j} : v + d^j \geq 0, \ \prod_{i=1}^{n_j} (v + d^j)^{w_i} \geq \alpha_j \right\}. \tag{37}$$

The set $\mathcal{A}_j$ identifies a weighted geometric mean inequality that must hold for the j-th CFMM. We include the nonnegative δ_j to account for noisy data. Choosing $\delta_j > 0$ gives a buffer for ensuring a transaction is still valid for noisy d (at the cost of reducing the achievable utility U).

Remark 1. *Ideally, we would directly compute $P_{\mathcal{A}_j}(x)$ in an algorithm computing optimal trades, which can be derived following an example in Beck's text.[14] However, this projection introduces unscalable coupling since, using $\delta_j = 0$ and $d = r$,*

$$[P_{\mathcal{A}_j}(x)]_i = \begin{cases} x_i & \textit{if } x \in \mathcal{A}_j \\ \dfrac{x_i - r_i^j + \sqrt{(x_i + r_i^j)^2 + 4\lambda w_i^j}}{2} & \textit{otherwise,} \end{cases} \tag{38}$$

where $\lambda > 0$ is a solution to

$$\sum_{i=1}^{n_j} w_i^j \log\left(\frac{x_i - r_i^j + \sqrt{(x_i + d_i^j)^2 + 4\lambda w_i^j}}{2}\right) = \log \alpha. \tag{39}$$

As the number of asset types in CFMMs increase, the time of a root finding algorithm to estimate λ also increases. Our alternative approach avoids this scaling issue. We also note JFB would technically require backpropping through the root finding scheme, but we suspect this could be avoided (by not attaching gradients during root finding) without adverse results.

Upon taking logarithms, we may equivalently write

$$\mathcal{A}_j = \left\{ v : v + d^j \geq 0, \ \sum_{i=1}^{n_j} w_i^j \ln(v_i + d_i^j) \geq \ln(\alpha_j) \right\} \tag{40a}$$

$$= \left\{ v : v + d^j \geq 0, \ \left\langle w, \ln(v + d^j) \right\rangle \geq \ln(\alpha_j) \right\}. \tag{40b}$$

[9]Note here we use the *backwards* of the referenced work, mapping global to local rather than local to global.

We decouple the constraint $\mathcal{A}_j$ by defining the hyperplane

$$\mathcal{H}_j \triangleq \left\{z : \left\langle w^j, z\right\rangle = \ln(\alpha_j)\right\}. \tag{41}$$

and the element-wise logarithm inequality constraint set

$$\mathcal{P}_j \triangleq \left\{(v, z) : z \leq \ln(v + d^j), v + d^j \geq 0\right\}. \tag{42}$$

These definitions yield the equivalence

$$v \in \mathcal{A}_j \quad \iff \quad \exists\, z \in \mathcal{H}_j \text{ s.t. } (v, z) \in \mathcal{P}_j. \tag{43}$$

This equivalence is useful since, as shown in a subsection below, $\mathcal{H}_j$ and $\mathcal{P}_j$ admit "nice" projection formulas. If instead the j-th CFMM is defined using a weighted arithmetic sum, then

$$\begin{aligned} \mathcal{A}_j &= \left\{x : x + d^j \geq 0, \left\langle w, x + d^j\right\rangle \geq \left\langle w, \hat{d}^j\right\rangle\right\} && \text{(44a)}\\ &= \left\{x : x + d^j \geq 0, \langle w, x\rangle \geq \left\langle w, \delta^j \odot d^j\right\rangle\right\}. && \text{(44b)}\end{aligned}$$

Next define the Cartesian product

$$\mathcal{A} = \mathcal{A}_1 \times \cdots \times \mathcal{A}_m. \tag{45}$$

This enables the constraints to be expressed by

$$\mathcal{C}_\Theta(d) = \{(x, y) \geq 0 : A^j(\gamma_j x^j - y^j) \in \mathcal{A}_j \;\forall\, j \in [m]\}. \tag{46}$$

The tunable weights in $\mathcal{C}_\Theta(d)$ consist of the constraint tolerances δ_j. Let us introduce an auxiliary variable z and the block diagonal matrix $A = \text{diag}(A^1, \ldots, A^m)$. Additionally, let $\mathcal{I}_1 \subset [m]$ be the subset of CFMM indices with weighted geometric product constraints and $\mathcal{I}_2 \triangleq [m] - \mathcal{I}_1$ the remaining indices for weighted sum constraints. We obtain feasibility if and only if (v, x, y, z) is a minimizer of the sum of indicator functions

$$\begin{aligned} &\delta_{\geq 0}(x) + \delta_{\geq 0}(y) + \delta_{\mathcal{R}}(v, x, y) && \text{(47a)}\\ &+ \sum_{j \in \mathcal{I}_1} \delta_{\mathcal{P}_j}(v^j, z^j) + \delta_{\mathcal{H}_j}(z^j) + \sum_{j \in \mathcal{I}_2} \delta_{\mathcal{A}_j}(v^j), && \text{(47b)}\end{aligned}$$

where

$$\mathcal{R} \triangleq \{(v, x, y) : v = \Gamma A x - A y\}. \tag{48}$$

This formulation of the constraints will be used in our operator splitting scheme.

Proximal/Gradient Operations

This section provides explicit formulas for the proximal and gradient operations needed. First note

$$P_{\geq 0}(x) = [x]_+ \triangleq \max(x, 0), \tag{49}$$

where the maximum occurs element-wise. The projection onto a hyperplane $\mathcal{H}_j$ is given by

$$P_{\mathcal{H}_j}(z) = z - \frac{\left\langle w^j, z - \ln(\hat{d}^j)\right\rangle}{\|w^j\|^2} w^j. \tag{50}$$

Similarly, if the j-th CFMM uses a weighted arithemtic,

$$P_{\mathcal{A}_j}(z) = z - \frac{[\langle w^j, z - \delta^j \odot d^j\rangle]_-}{\|w^j\|^2} w^j, \tag{51}$$

where $[z]_- \triangleq \min(z, 0)$. Next, the projection $P_{\mathcal{P}_j}$ is defined element-wise. The element-wise slope of $\ln(v + d)$ is $1/(v + d)$. The negative reciprocal of the slope (*i.e.* $-(v + d)$) gives the slope of the normal line passing through the projection and the point of interest. Letting $(\overline{v}^j, \overline{z}^j)$ be the projection of (v^j, z^j) gives the point-slope relation

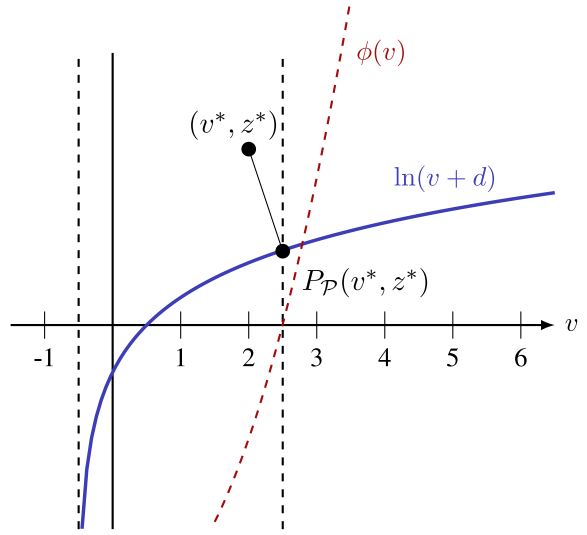

Figure 11: Illustration for projection in $\mathbb{R}^2$ onto the set $\mathcal{P} \triangleq \{(v^*, z^*) : \ln(v + d) \geq z, v + d \geq 0\}$, which is all points below the blue curve $\ln(v + d)$. Here $d = 1/2$. The dashed red curve shows $\phi(v)$, the function defining the optimality condition for the projection in (53).

$$\overline{z}^j - z^j = -(\overline{v}^j + d) \odot (\overline{v}^j - v^j). \tag{52}$$

Defining the function

$$\phi(v) \triangleq v \odot v + v \odot (d - v^j) - d \odot v^j + \ln(v + d) - z^j \tag{53}$$

enables the relation (52) can be expressed as

$$\phi(\overline{v}^j) = 0. \tag{54}$$

Since the above relation is element-wise and separable, each component $\overline{v}_i^j$ can be found independently (*e.g.* via a Newton iteration). We emphasize solving for each $\overline{v}_i^j$ is independent of the dimension n_j whereas computation costs for λ in (39) *increase* with n_j.

The final projection is for the linear constraint $\mathcal{R}$. The projection $P_{\mathcal{R}}(v, x, y)$ is a solution to the problem

$$\min_{(\overline{v}, \overline{x}, \overline{y})} \|\overline{v} - v\|^2 + \|\overline{x} - x\|^2 + \|\overline{y} - y\|^2 \;\text{ s.t. }\; \overline{v} = A(\Gamma\overline{x} - \overline{y}). \tag{55}$$

Let $N = [\Gamma A - A]$ and $\overline{q} = (\overline{x}, \overline{y})$ so the problem becomes

$$\min_{(\overline{v}, \overline{q})} \|\overline{v} - v\|^2 + \|\overline{q} - q\|^2 \;\text{ s.t. }\; \overline{v} = N\overline{q}. \tag{56}$$

It suffices to solve for $\overline{q}$ since the optimal $\overline{v}$ is then obtained by applying N. Substituting this in yields the simpler problem

$$\min_{\overline{q}} \|N\overline{q} - v\|^2 + \|\overline{q} - q\|^2, \tag{57}$$

for which the optimality condition is

$$0 = N^\top(N\overline{q}^\star - v) + \overline{q}^\star - q. \tag{58}$$

Rearranging gives the formula

$$\overline{q}^\star = (\mathrm{I} + N^\top N)^{-1}(q + N^\top v). \tag{59}$$

Letting

$$M \triangleq (I + N^\top N)^{-1} \tag{60}$$

and substituting in for $N^\top$ reveals

$$[P_{\mathcal{R}}(v,x,y)]_{(x,y)} = M \begin{bmatrix} x + A^\top \Gamma v \\ y - A^\top v \end{bmatrix}. \tag{61}$$

Lastly, we express the gradient for the utility U_Θ. Here

$$U_\Theta(x,y) = \sum_{j=1}^{m} \left\langle A^j p, A^j(y^j - x^j) \right\rangle \tag{62a}$$

$$- \frac{1}{2}\|W^j A^j (y^j - x^j)\|^2, \tag{62b}$$

where A^j is used to ensure the utility only measures cryptoassets that are available on the j-th CFMM (*i.e.* converts the global coordinates of x^j and y^j into the local coordinates of the CFMM), and each $W^j \in \mathbb{R}^{n_j \times n_j}$ penalizes transaction sizes in the j-th CFMM. For each j,

$$\nabla_{x^j} U_\Theta = -p - (W^j A^j)^\top (W^j A^j)(x^j - y^j) \tag{63}$$

and

$$\nabla_{y^j} U_\Theta = -\nabla_{x^j} U_\Theta. \tag{64}$$

Furthermore, U_Θ is L-Lipschitz with

$$L \triangleq \max_{j \in [m]} \|W^j A^j\|_2. \tag{65}$$

Operator Splitting Formulation

Set $\xi = (v, x, y, z)$ and define the functions

$$\delta_{\mathcal{M}}(v,z) \triangleq \sum_{j \in \mathcal{I}_1} \delta_{\mathcal{P}_j}(v^j, z^j) + \sum_{j \in \mathcal{I}_2} \delta_{\mathcal{A}_j}(v^j), \tag{66a}$$

$$\delta_{\mathcal{H}}(z) \triangleq \sum_{j \in \mathcal{I}_1} \delta_{\mathcal{H}_j}(z^j), \tag{66b}$$

where $\mathcal{M}$ and $\mathcal{H}$ are the sets corresponding to where the indicators in their definitions are all zero. Then define the functions

$$f(\xi) \triangleq \delta_{\geq 0}(x,y) + \delta_{\mathcal{M}}(v,z), \tag{67a}$$

$$g(\xi) \triangleq \delta_{\mathcal{R}}(v,x,y) + \delta_{\mathcal{H}}(z) \tag{67b}$$

$$h(\xi) \triangleq -U_\Theta(x,y). \tag{67c}$$

The problem (20) may be equivalently expressed by

$$\min_{\xi} f(\xi) + g(\xi) + h(\xi), \tag{68}$$

where we note f and g are proximable and h is L-Lipschitz differentiable. We use Davis-Yin splitting[22] and $\alpha > 0$ iterate via

$$\xi^{k+1} = \text{prox}_{\alpha f}(\zeta^k) \tag{69a}$$

$$\psi^{k+1} = \text{prox}_{\alpha g}(2\xi^{k+1} - \zeta^k - \alpha \nabla h(\xi^{k+1})) \tag{69b}$$

$$\zeta^{k+1} = \zeta^k + \psi^{k+1} - \xi^{k+1}. \tag{69c}$$

For step size $\alpha \in (0, 2/L)$, we obtain the desired convergence $(\xi_x^{k+1}, \xi_y^{k+1}) \to (x_d, y_d)$.