Hysteretic neural networks and trading

March 23, 2018

1 Hysteretic neural networks

1.1 Play and Prandtl–Ishlinskii networks

Consider $K > 0$ play operators. Each of them maps an initial state $p_0^k \in \mathbb{R}$ and an input sequence $x_1, x_2, \dots$ to an output sequence $p_1^k, p_2^k, \dots$, i.e.,

$$p_0^k, (x_1, x_2, \dots) \mapsto (p_1^k, p_2^k, \dots), \quad k = 1, \dots, K.$$

The kth play operator is given by:

$$p_n^k = G(x_n, p_{n-1}^k, w_1^k) := p_{n-1}^k + \Phi(w_1^k x_n - p_{n-1}^k), \quad n = 1, 2, \dots, \tag{1.1}$$

where w_1^k are parameters and $\Phi(x) = x$ for $x > 0$, $\Phi(x) = x + 1$ for $x < -1$ and $\Phi(x) = 0$ for $x \in [-1, 0]$, see Fig. 1.1.

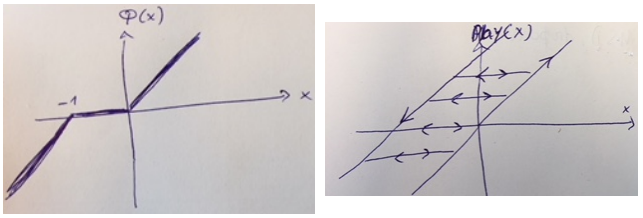


Figure 1.1: Left: Function $\Phi(x)$. Right: Play operator.

It can be represented as a recurrent neural network, see Fig. 1.2 (left). Note that in such a form the network is not feed-forward. One can unfold it to make it feed-forward, see Fig. 1.2 (right).

Definition 1.1. We call this network a *play network*. If there are m elements in the sequence $\{x_n\}$, we say the unfolded network is *m-unfolded*.

For example, the network in Fig. 1.2 (right) is 2-unfolded.

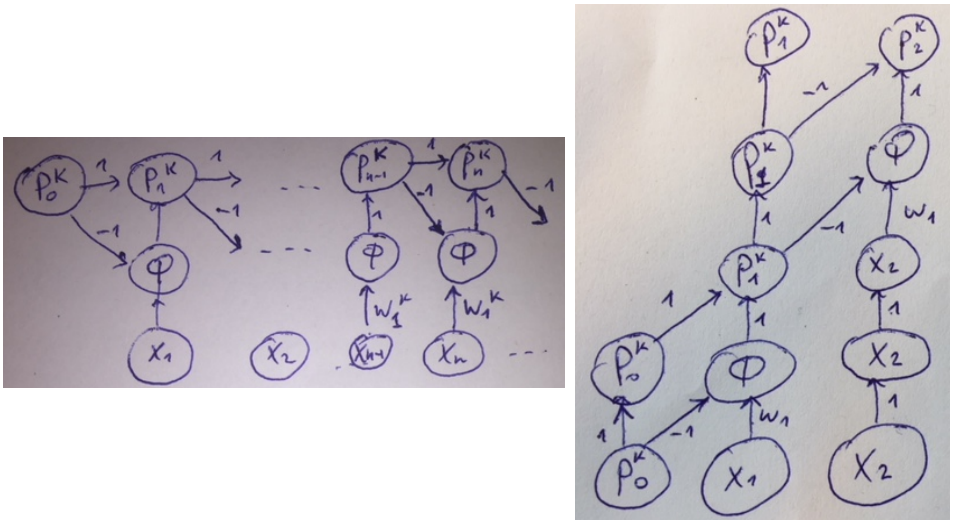


Figure 1.2: Left: Recurrent play network. Right: Unfolded play network.

1.2 Play layers and Prandtl–Ishlinskii networks

Denote a vector of initial states by

$$P_0 := (p_0^1, \dots, p_0^K) \in \mathbb{R}^K.$$

The Prandtl–Ishlinskii (PI) operator maps

$$P_0, (x_1, x_2, \dots) \mapsto (p_1, p_2, \dots)$$

and is given by:

$$p_n = \sum_{k=1}^{K} w_2^k p_n^k + w_3 = \sum_{k=1}^{K} w_2^k G(x_n, p_{n-1}^k, w_1^k) + w_3, \quad n = 1, 2, \dots \tag{1.2}$$

where w_2^k, w_3 are parameters and p_n^k are given by (1.1). It corresponds to a layer of plays or a *PI network*, see Fig. 1.3. It can be unfolded similarly to Fig. 1.2 (right).

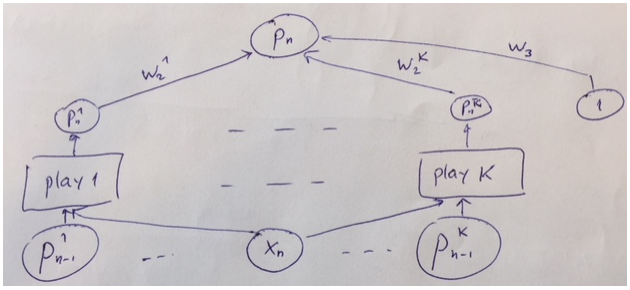


Figure 1.3: A play layer.

Remark 1.1. What if some of the weights become negative?
Should we add linear neurons to the layer of Φ's in Fig. 1.2 (left)?
What is a good approximation of a general Preisach operator?

1.3 Training a PI network

Assume we are given an input sequence $x_1, x_2, \ldots, x_N$ and an output sequence $q_1, q_2, \ldots, q_N$. We perform the following steps in cycle until convergence.

1. Preparing initial states for the m-unfolded network: Fix a vector of initial states P_0 and all the weights (denoted by W). For each $k = 1, \ldots, K$, we calculate recursively $p_1^k, p_2^k, \ldots, p_N^k$ by formula (1.1). We denote the corresponding (intermediate) states of the PI operator by
$$P_n = (p_n^1, \ldots, p_n^K), \quad n = 1, \ldots, N.$$

2. Preparing inputs for the m-unfolded network: We fix m and group the input sequence into m-tuples:
$$\mathbf{x}_1 := (x_1, \ldots, x_m), \quad \mathbf{x}_2 := (x_2, \ldots, x_{m+1}), \ldots,$$
which gives $M := N - m$ tuples $\mathbf{x}_1, \ldots, \mathbf{x}_M$. Next we form a new set of inputs for the m-unfolded network, attaching the vectors of intermediate states:
$$\mathbf{y}_1 := (P_0, \mathbf{x}_1), \quad \mathbf{y}_2 := (P_1, \mathbf{x}_2), \ldots,$$
where P_0 was fixed at Step 1 and $P_1, P_2, \ldots$ were recursively calculated at Step 1.

3. Training the m-unfolded network: We train by stochastic gradient descent the feed-forward m-unfolded PI network
$$\mathbb{R}^K \times \mathbb{R}^m \ni \mathbf{y} \mapsto F_m(\mathbf{y}) \in \mathbb{R}^m$$
with the inputs $\mathbf{y}_1, \ldots, \mathbf{y}_M$ and the true targets $\mathbf{q}_1, \ldots, \mathbf{q}_M$, where
$$\mathbf{q}_1 = (q_1, \ldots, q_m), \quad \mathbf{q}_2 = (q_2, \ldots, q_{m+1}), \ldots$$
At this step, we only update the weights W, but not the initial state P_0. For example, we can train for one epoch.

4. We update the initial state P_0:
$$P_0^{\text{new}} := P_0 - \nabla_{P_0} \left(F_m(P_0, \mathbf{x}_1) - \mathbf{q}_1 \right)^2.$$

1.4 General hysteretic networks

A general network may consist of several play layers (and perhaps standard layers). We call such a network *hysteretic*, and denote
$$p_n = F(X_n, W), \tag{1.3}$$
where $X_n = (x_1, \ldots, x_n)$ and w is a vector of all weights of the network.

2 Trading

2.1 Dima's model assumptions

Assume, we observe prices $p_1, \dots, p_N$ for a fixed $N > 0$. Based on Dima's paper [1], assume that the price p_n hysteretically depends on the underlying noise b_n, with b_n being a Brownian motion. Using the notation

$$\mathcal{B}_n := (b_1, \dots, b_n), \quad \mathcal{P}_n := (p_1, \dots, p_n),$$
$$\mathcal{B} := \mathcal{B}_N, \quad \mathcal{P} := \mathcal{P}_N,$$

we have

$$b_0 = 0, \quad b_n \sim \mathcal{N}(b_{n-1} + \mu_b, \tau_b^{-1}), \tag{2.1}$$
$$p_n = F(\mathcal{B}_n, W_p), \tag{2.2}$$

where F is a hysteresis operator parametrized by a vector W_p. The vector W_p contains the weights of the network and the vector of initial states of all the individual play operators.

Based on Dima's paper [1] again, the underlying noise b_n can be expressed as a hysteresis operator depending on the (observed) prices p_n, i.e.,

$$b_n = G(\mathcal{P}_n, W_b), \tag{2.3}$$

where G is a hysteresis operator parametrized by a vector W_b. The vector W_b contains the weights of the network and the vector of initial states of all the individual play operators.

This is very nice as long as F and G are Prandtl–Ishlinskii operators. However, if one explicitly adds N's strategy, G becomes Preisach and F is not Preisach anymore. It is not clear how well it can be approximated by compositions of plays and nonlinear functions???

2.2 Direct Learning

We learn the parameters W_b, μ_b, τ_b and the initial state P_0 of the network G by maximizing the likelihood of $\mathcal{P}$. Since $\mathcal{P}$ is the determenistic function (2.2) of a random variable $\mathcal{B}$, its probability density is given by

$$p(\mathcal{P}) = p_b(G(\mathcal{P}_1, W_b), \dots, G(\mathcal{P}_N, W_b))|\det J(\mathcal{P})|, \tag{2.4}$$

where

$$p_b(\mathcal{B}) = \prod_{n=1}^{N} p_b(b_n|b_{n-1}) = \prod_{n=1}^{N} \frac{\tau_b^{1/2}}{\sqrt{2\pi}} e^{-(b_n - b_{n-1} - \mu_b)^2 \tau_b/2} \tag{2.5}$$

is the probability distribution of $\mathcal{B}$ and $J(\mathcal{P})$ is the Jacobian matrix. Recall that G has the causality property, hence $J(\mathcal{P})$ is a triangular matrix. Therefore, (2.4) and (2.5) yield

$$p(\mathcal{P}) = \prod_{n=1}^{N} \left[\frac{\tau_b^{1/2}}{\sqrt{2\pi}} e^{-(G(\mathcal{P}_n, W_b) - G(\mathcal{P}_{n-1}, W_b) - \mu_b)^2 \tau_b/2} \left| \frac{\partial G(\mathcal{P}_n, W_b)}{\partial p_n} \right| \right].$$

Thus, maximizing the log-likelihood of $p(\mathcal{P})$ is equivalent to the following:

$$-\frac{1}{2}\sum_{n=1}^{N}\left((G(\mathcal{P}_n,W_b)-G(\mathcal{P}_{n-1},W_b)-\mu)^2\tau_b-\ln\tau_b\right)\left|\frac{\partial G(\mathcal{P}_n,W_b)}{\partial p_n}\right|\to\max_{W_b,\mu_b,\tau_b,P_0}. \tag{2.6}$$

We can maximize it analogously to Sec. 1.3, using the m-unfolded network. As a by-product, we obtain a determenistic sequence $b_1,\dots,b_N$ of the noise levels, which means that $p(\mathcal{B}|\mathcal{P})$ is a delta-distribution.

The inverse Preisach $F(\mathcal{B},W_p)$ can be approximated by another network (trained as in Sec. 1.3 with the given inputs $b_1,\dots,b_N$ obtained above and given ground truth outputs $p_1,\dots,p_N$). **It is problematic if G is not Prandtl–Ishlinskii. Then we should use the methods in Sec. 3.2.???**

2.3 Latent variable formulation and expectation-maximization algorithm

2.3.1 Latent variable formulation

We assume that $\mathcal{P}=(p_1,\dots,p_n)$ (price) and $\mathcal{B}=(b_1,\dots,b_N)$ (noise, latent variables) are random variables with the joint distribution

$$\begin{aligned}&p(\mathcal{P},\mathcal{B}|\theta)=\prod_{n=1}^{N}p(p_n|\mathcal{B}_n,W_p,\tau_p)p(b_n|b_{n-1},\mu_b,\tau_b),\\&p(p_n|\mathcal{B}_n,W_p,\tau_p)=\mathcal{N}(F(\mathcal{B}_n,W_p),\tau_p^{-1}),\quad p(b_n|b_{n-1},\mu_b,\tau_b)=\mathcal{N}(b_{n-1}+\mu_b,\tau_b^{-1}),\end{aligned} \tag{2.7}$$

where $\theta=(W_p,\tau_p,\mu_b,\tau_b)$ are unknown parameters. Note that we relaxed assumption (2.2) by introducing a variance τ_p^{-1}.

Our goal is to find the parameters θ by maximizing the likelihood function

$$p(\mathcal{P}|\theta)=\int p(\mathcal{P},\mathcal{B}|\theta)\,d\mathcal{B}\to\max_{\theta}. \tag{2.8}$$

2.3.2 Learning: general expectation-maximization algorithm

Since the integral in (2.8) has no closed form, we will use the expectation-maximization (EM) algorithm [3]. It is based on the identity

$$\ln p(\mathcal{P}|\theta)=Q(q,\theta)+E(q)+KL(q\|p,\theta), \tag{2.9}$$

where

$$Q(q,\theta):=\int q(\mathcal{B})\ln p(\mathcal{P},\mathcal{B}|\theta)\,d\mathcal{B}, \tag{2.10}$$

$$E(q):=-\int q(\mathcal{B})\ln q(\mathcal{B})\,d\mathcal{B}, \tag{2.11}$$

$$KL(q\|p,\theta) := \int q(\mathcal{B}) \ln\left(\frac{q(\mathcal{B})}{p(\mathcal{B}|\mathcal{P},\theta)}\right), \tag{2.12}$$

which holds for any probability distribution $q(z)$.

The algorithm alternatively fixes $q(\mathcal{B}) = p(\mathcal{B}|\mathcal{P},\theta)$ (which makes the KL-divergence vanish) and maximizes $Q(q,\theta)$ (or equivalently $Q(q,\theta) + E(q)$) with respect to θ. Let us write it in more detail in our situation.

1. We choose an initial setting for the parameters θ^{old}.

2. **E step:** We evaluate the posterior distribution

$$p(\mathcal{B}|\mathcal{P},\theta^{\text{old}}) = \frac{p(\mathcal{P},\mathcal{B}|\theta^{\text{old}})}{C(\mathcal{P},\theta^{\text{old}})}, \tag{2.13}$$

where $C(\mathcal{P},\theta^{\text{old}})$ is a normalization constant. We define the expectation function

$$Q(\theta,\theta^{\text{old}}) := \int \ln p(\mathcal{P},\mathcal{B}|\theta) p(\mathcal{B}|\mathcal{P},\theta^{\text{old}})\, d\mathcal{B}. \tag{2.14}$$

This integral is still not feasible from the practical point of view. Therefore, we will approximate $p(\mathcal{B}|\mathcal{P},\theta^{\text{old}})$ by a delta function supported at the maximum of $p(\mathcal{B}|\mathcal{P},\theta^{\text{old}})$:

$$p(\mathcal{B}|\mathcal{P},\theta^{\text{old}}) \approx \delta(\mathcal{B} - \mathcal{B}^{\max}) = \prod_{n=1}^{N} \delta(b_n - b_n^{\max}), \tag{2.15}$$

where $\mathcal{B}^{\max} = \mathcal{B}^{\max}(\theta^{\text{old}})$ is given by

$$\mathcal{B}^{\max} = (b_1^{\max},\dots,b_N^{\max}) = \arg\max_{\mathcal{B}} p(\mathcal{P},\mathcal{B}|\theta^{\text{old}}). \tag{2.16}$$

This approximation allows us to not calculate the normalization constant $C(\mathcal{P},\theta^{\text{old}})$ in (2.13) and to not evaluate the integral in (2.14). Instead, we approximate the function $Q(\theta,\theta^{\text{old}})$ by

$$\tilde{Q}(\theta,\theta^{\text{old}}) := \ln p(\mathcal{P},\mathcal{B}^{\max}(\theta^{\text{old}})|\theta). \tag{2.17}$$

3. **M step:** We find θ^{new} given by

$$\theta^{\text{new}} := \arg\max_{\theta} \tilde{Q}(\theta,\theta^{\text{old}}). \tag{2.18}$$

4. We check for convergence. If the convergence criterion is not satisfied, then we set

$$\theta^{\text{old}} := \theta^{\text{new}}$$

and return to step 2.

As a by-product of the EM algorithm, we get a sequence $b_1^{\max},\dots,b_N^{\max}$ of most probable values of noise. Now we can train the network (2.3) as described in Sec. 1.3.

2.3.3 Learning: practical implementation

To implement the EM algorithm we have to perform the following maximizations.

E step. Due to (2.16), we have to maximize

$$p(\mathcal{P},\mathcal{B}|\theta^{\text{old}}) = -\frac{1}{2}\sum_{n=1}^{N}\Big[(p_n - F(\mathcal{B}_n, W_p^{\text{old}}))^2\tau_p^{\text{old}} - \ln\tau_p^{\text{old}} + (b_{n+1} - b_n - \mu^{\text{old}})^2\tau_b^{\text{old}} - \ln\tau_b^{\text{old}}\Big] \quad (2.19)$$

with respect to $b_1,\dots,b_N$. Unlike in Sec. 1.3, now the parameters of the network are fixed and we optimize the input. As in Sec. 1.3, we transform the network F into the m-unfolded network and prepare the m-tuples $\mathbf{b}_1,\mathbf{b}_2,\dots,\mathbf{b}_M$ (that we initialize randomly for the first time or from the previous iteration of the EM algorithm). Then we find the intermediate states $P_0,P_1,\dots,P_M$, where $P_0 = P_0^{\text{old}}$ and $P_1,P_2,\dots$ are evaluated recursively. After that, we have the inputs

$$(P_0,\mathbf{b}_1),(P_1,\mathbf{b}_2),\dots$$

for the m-unfolded network with fixed weights. Now, in cycle, for $j = 1,2\dots,M$, we update $\mathbf{b}_j$ by gradient ascent of (2.19) and recalculate P_j. We repeat until convergence.

M step. Due to (2.17) and (2.18), we have to maximize

$$\tilde{Q}(\theta,\theta^{\text{old}}) = -\frac{1}{2}\sum_{n=1}^{N}\Big[(p_n - F(\mathcal{B}_n^{\max}, W_p))^2\tau_p - \ln\tau_p + (b_{n+1}^{\max} - b_n^{\max} - \mu^{\text{old}})^2\tau_b - \ln\tau_b\Big] \quad (2.20)$$

with respect to $\theta = (W_p,\tau_p,\mu_b,\tau_b)$. Now the input sequence is given, and we can proceed analogously to Sec. 1.3.

2.4 Learning using the density functions

This learning method is based on the material in Sec. 3.2. We learn a function $S(x,y,W_S)$ that allows for an alternative representation of the hysteresis operator in the form

$$p_n = S(p_{n'},\Delta b, W_S), \quad \Delta b := b_n - b_{n-1}, \quad (2.21)$$

where $n' < n$ is uniquely determined by the sequence $p_1,\dots,p_n$. This representation is possible under the assumption that

$$\min_{j=1,\dots,n-1} p_j \le p_n \le \max_{j=1,\dots,n-1} p_j \quad (2.22)$$

(we omitted the dependencies on W_T, W_S). If this assumption does not hold, then n' also depends on the (unknown) initial state, and some modifications of the algorithm below are needed.

Along with the function S, we will learn a function $T(x,y,W_T)$ such that S is inverse to T with respect to y, i.e.,

$$S(x,T(x,y)) = y. \quad (2.23)$$

Suppose we are given a sequence of (observed) prices $p_1,\dots,p_N$.

1. We initialize W_S, W_T.

2. Using the functions $T(x, y, W_T)$ and $S(x, y, W_S)$, we construct a subsequence (see Sec. 3.2)
$$p_{i_1}, \dots, p_{i_K} = p_{N-1} \tag{2.24}$$
and assume that the noise b_n satisfies the relations
$$\Delta b = T(p_{i_{k-1}}, p_{i_k}, W_T), \tag{2.25}$$
which would hold if T and S were the correct functions from Sec. 3.2. **Dima, how should we define p_{i_0}?** Recalling that Δb is a random variable (see (2.1)), we see that relation (2.25) defines the joint distribution
$$\prod_{k=1}^{K} p(p_{i_k}|p_{i_{k-1}}) = \prod_{k=1}^{K} \left[\frac{\tau_b^{1/2}}{\sqrt{2\pi}} e^{-(T(p_{i_{k-1}}, p_{i_k}, W_T) - \mu_b)^2 \tau_b / 2} \left| \frac{\partial T(p_{i_{k-1}}, p_{i_k}, W_T)}{\partial p_{i_k}} \right| \right], \tag{2.26}$$
cf. (2.4). Using the observed prices (2.24), we maximize the log-likelihood of (2.26) with respect to W_T, τ_b, μ_b.

3. We update the weights W_S by maximizing (see (2.23))
$$\sum_{x,y} \left(S(x, T(x, y)) - y \right)^2 \to \min_{W_S}. \tag{2.27}$$
What x, y should we sum over? Over p_{i_k}? Over p_n? Over all possible x, y?

4. If a convergence criterion does not hold, go to step 2.

As a result, we obtain the network approximation of (2.21) and the parameters μ_b, τ_b of the noise increment Δb. This network can be used for predicting next price.

2.5 Predicting

For the methods described in Sections 2.2 and 2.3, we predict the next price:
$$p_{N+1}^{\text{predicted}} = F((B_N, b_N + \Delta b), W_p) + \mathcal{N}(0, \tau_p), \quad \Delta b \sim \mathcal{N}(\mu_b, \tau_b),$$
e.g., by sampling or by propagating $\mathcal{N}(\mu, \tau)$ through the network similarly to [2].

Once we observed the true price p_{N+1}, we predict the next noise level:
$$b_{N+1} = G(P_{N+1}, W_b).$$
We keep predicting the next price and the next noise level in turn.

For the method described in Sec. 2.4, we predict the next price directly by
$$p_{N+1}^{\text{predicted}} = T(p_{N'}, \Delta b, W_T), \quad \Delta b \sim \mathcal{N}(\mu_b, \tau_b).$$

3 Preisach model and its variants

3.1 The Preisach model

For our purposes, it suffices to consider the Preisach model of the form

$$p_i = a\, x_n + \int_0^\infty g_r(r, \mathrm{Play}_r[x_i])\, dr,$$

or, to cover a discrete case,

$$p_i = \int_0^\infty g_r(r, \mathrm{Play}_r[x_i])\, d\mu(r).$$

- In D's model, an individual agent can be approximated by the difference of two plays with almost identical thresholds:

  $$S[x_i] = \frac{1}{\varepsilon}\big(\mathrm{Play}_r[x_i] - \mathrm{Play}_{r+\varepsilon}[x_i]\big).$$

  The aggregation of these agents leads to the Prandt-Ishlinskii operator

  $$p_i = \int_0^\infty \mathrm{Play}_r[x_i]\, d\mu(r),$$

  which is a particular case of the Preisach model.

- In N's model, the trading strategy leads the Preisach model again, now with a negative function $g(\cdot,\cdot)$. We need to discuss the typical profile of the nonlinearity $g(\cdot,\cdot)$ for this model.

- Combining D's and N's agents results in the Preisach operator again (a linear combination of Presach operators).

3.2 An Algorithmic Description with Everett's Function

Suppose p_i and b_i, $i = 0, 1, \dots$ are an input sequence and output sequence of the Preisach model, respectively. We also consider the increments $\Delta p_i = p_i - p_{i-1}$ and $\Delta b_i = b_i - b_{i-1}$.

Take an $N \geq 0$ and the corresponding p_N. If either $p_N \geq p_0, p_1 \dots, p_{N-1}$ or $p_N \leq p_0, p_1 \dots, p_{N-1}$, then we don't use this data point.

Otherwise, we associate with p_N a subsequence p_{i_k} $(0 \leq i_0 < i_1 < \dots < i_K = N-1)$ of the previous data points by the following rules. Note that the indices i_k and the number K depend on N.

If $p_{N-1} < p_N$, then:

- First, find p_{i_0} with $i_0 < N$ such that

  $$p_{i_0} > p_N, \qquad p_{i_0+1}, p_{i_0+2}, \dots, p_{N-1} < p_N.$$

- Find the minimum of the input values $p_{i_0}, p_{i_0+1}, \ldots, p_{N-1}$, with the corresponding moment when it is achieved, and denote this p_{i_1}:

$$p_{i_1} = \min\{p_{i_0}, p_{i_0+1}, \ldots, p_{N-1}\}.$$

- Next, we define recurrently

$$p_{i_2} = \max\{p_{i_1}, p_{i_1+1}, \ldots, p_{N-1}\},$$

$$p_{i_3} = \min\{p_{i_2}, p_{i_2+1}, \ldots, p_{N-1}\},$$
$$p_{i_4} = \max\{p_{i_3}, p_{i_3+1}, \ldots, p_{N-1}\},$$
$$p_{i_5} = \min\{p_{i_5}, p_{i_5+1}, \ldots, p_{N-1}\},$$
$$\vdots$$

  The last point in this sequence will be p_{N-1}, i.e. $i_K = N-1$. In particular,

$$p_{i_1} < p_{i_3} < \cdots < p_{i_K} = p_{N-1} < \cdots < p_{i_4} < p_{i_2} < p_N < p_{i_0}.$$

- With this sequence at hand,

$$\Delta b_N = T(p_{i_1}, p_N) - \sum_{k=1}^{K-1} T(i_k, i_{k+1}), \tag{3.1}$$

  where the function $T(\cdot,\cdot)$ satisfies

$$T(x, y) = -T(y, x).$$

The function $T(\cdot,\cdot)$ is to be identified. The corresponding term (factor) in the likelihood function is:

$$\exp\bigl(-(\Delta b_N)^2/2\bigr)\, \frac{\partial T}{\partial y}(p_{i_1}, p_N).$$

Similarly, if $p_{N-1} > p_N$, then:

- We find p_{i_0} with $i_0 < N$ such that

$$p_{i_0} < p_N, \qquad p_{i_0+1}, p_{i_0+2}, \ldots, p_{N-1} > p_N.$$

- Then, find the maximum of the input values $p_{i_0}, p_{i_0+1}, \ldots, p_{N-1}$, with the corresponding moment when it is achieved, and denote this p_{i_1}:

$$p_{i_1} = \max\{p_{i_0}, p_{i_0+1}, \ldots, p_{N-1}\}.$$

- Next, define recurrently

$$p_{i_2} = \min\{p_{i_1}, p_{i_1+1}, \dots, p_{N-1}\},$$

$$p_{i_3} = \max\{p_{i_2}, p_{i_2+1}, \dots, p_{N-1}\},$$
$$p_{i_4} = \min\{p_{i_3}, p_{i_3+1}, \dots, p_{N-1}\},$$
$$p_{i_5} = \max\{p_{i_5}, p_{i_5+1}, \dots, p_{N-1}\},$$
$$\vdots$$

The last point in this sequence will be, again, p_{N-1}, i.e. $i_K = N-1$, and

$$p_{i_1} > p_{i_3} > \cdots < p_{i_K} = p_{N-1} > \cdots > p_{i_4} > p_{i_2} > p_N > p_{i_0}.$$

- The increment Δb_N is defined by the same formula (3.1).

The corresponding term (factor) in the likelihood function is:

$$\exp\bigl(-(\Delta b_N)^2/2\bigr)\,\frac{\partial T}{\partial x}(p_{i_1}, p_N)$$

(the other derivative).

3.3 The Prandtl–Ishlinskii model

This is a particular case of the Preisach model with Everett's function of the special form

$$T(x, y) = \Pi(y - x),$$

which is effectively a function of one variable.

3.4 The Inverse Preisach model

We consider the Preisach model but now the output is known and the input is not. If The function $T(x, y)$ *strictly increases* in y, then the algorithm presented in Section 3.2 can be adjusted in a very simple way. Namely, all the elements x_n in the bullet points should be replaced with p_n, and the counterpart of (3.1) should read

$$x_{i_k} = T^{-1}(x_{i_{k-1}}, p_{i_k} - p_{i_{k-1}}), \qquad k = 1, \dots, K, \tag{3.2}$$

where the inversion is with respect to the second variable. Consecutive substitutions then give

$$x_N = T^{-1}(T^{-1}(\dots), p_N - p_{i_{K-1}}). \tag{3.3}$$

4 The Moving Preisach Model

Our financial model can be written as

$$p_n = \text{Preisach}\,[p_n] + x_n.$$

This is known as the Moving Preisach Model in the deterministic setting. A numerical scheme for the case where x_n is the Brownian motion was considered in [4].

This equation covers both trading strategies of N and D, and their mixture. Brokate shows that under some assumptions this model has the *wiping out property*, see [5]. Since the identity operator minus the Preisach operator is a Preisach operator, then the equation is equivalent to

$$\hat{\text{P}}\text{reisach}\,[p_n] = x_n, \qquad p_n = \hat{\text{P}}\text{reisach}^{-1}\,[x_n],$$

where $\hat{\text{P}}\text{reisach} = \mathbb{I} - \text{Preisach}$ and we assume that this Preisach operator is invertible.

The invertibility holds true for the N's strategy because in this case the Preisach density of Preisach is negative.

The invertibility also holds true if D's strategy is used (one should be careful with uniqueness though) because in this case the operator Preisach is monotone (and bounded).

If the strategies are mixed then the invertibility also seems to hold because

$$p_n = \text{Preisach}_N\,[p_n] + \text{Preisach}_D\,[p_n] + x_n$$

is equivalent to

$$p_n = \big(\mathbb{I} - \text{Preisach}_N\big)^{-1}\big(\text{Preisach}_D\,[p_n] + x_n\big),$$

where the inverse in the right hand side and the operator Preisach_D are both monotone (this needs more thinking).

4.1 Non-fixed Point Model

Another version of the model can be...

5 Second model: demand/supply price formation

In this model, we assume that the stock (or, e.g., bitcoin) exchange accommodates two types of agents, which we call D and N and describe below. We assume that the time takes values $n = 0, 1, 2, \ldots$, and denote the price at time n by p_n. By definition, p_n is the price of the last transaction at time n. We will see below that demand equals supply at times n, but between any two consecutive times $n-1$ and n, this need not be the case and there may occur many transactions until demand will have become supply.

Assumption 5.1 (Strategy of agents D)**.** *Agents D keep track of a trend. They buy stocks iff the price goes up and sell stocks iff the price goes down. The total amount of stocks that*

are in possession of all the agents D can be described as a Prandtl–Ishlinskii operator whose input is the price p. We denote this operator by

$$P_D(p).$$

Assumption 5.2 (Strategy of agents N)**.** *The strategy of each of the agents N is characterized by a non-ideal relay with two fixed thresholds $p_1 < p_2$ (different for different agents). The relay is in state 0 if the price is higher than p_2 and in state 1 if the price is lower than p_1. The agent buys one stock whenever his relay switches from 0 to 1 and sells one stock whenever his relay switches from 1 to 0. The total amount of stocks that are in possession of all the agents N can be described as a Preisach operator whose input is the price p. We denote this operator by*

$$P_N(p).$$

The following lemma is a direct consequence of the definition of the operators $P_D(p)$ and $P_N(p)$.

Lemma 5.1. *1. If p is increasing, then $P_D(p)$ is increasing (agents D buy) and $P_N(p)$ is decreasing (agents N sell).*

2. If p is decreasing, then $P_D(p)$ is decreasing (agents D sell) and $P_N(p)$ is increasing (agents N buy).

If p is increasing, then $P_D(p)$ is increasing (agents D buy) and $P_N(p)$ is decreasing (agents N sell).

Remark 5.1. Since $P_D(p)$ and $P_N(p)$ are not functions but operators, the monotone curves described in Lemma 5.1 are not defined only by the current value of p, but depend on its prehistory.

Denote by K_n the total amount of stocks that are in possession of all the agents D and N at time n. We will see below in Assumption 5.4 that K_n may change as time goes on.

Assumption 5.3 (Demand/supply price formation)**.** *At each moment n, the price p_n is a* stable solution *of the equation*

$$P_D(p_n) + P_N(p_n) = K_n, \tag{5.1}$$

where the stability notion is explained in Fig. 5.1. We will say that p_n is a stabilized price*.*

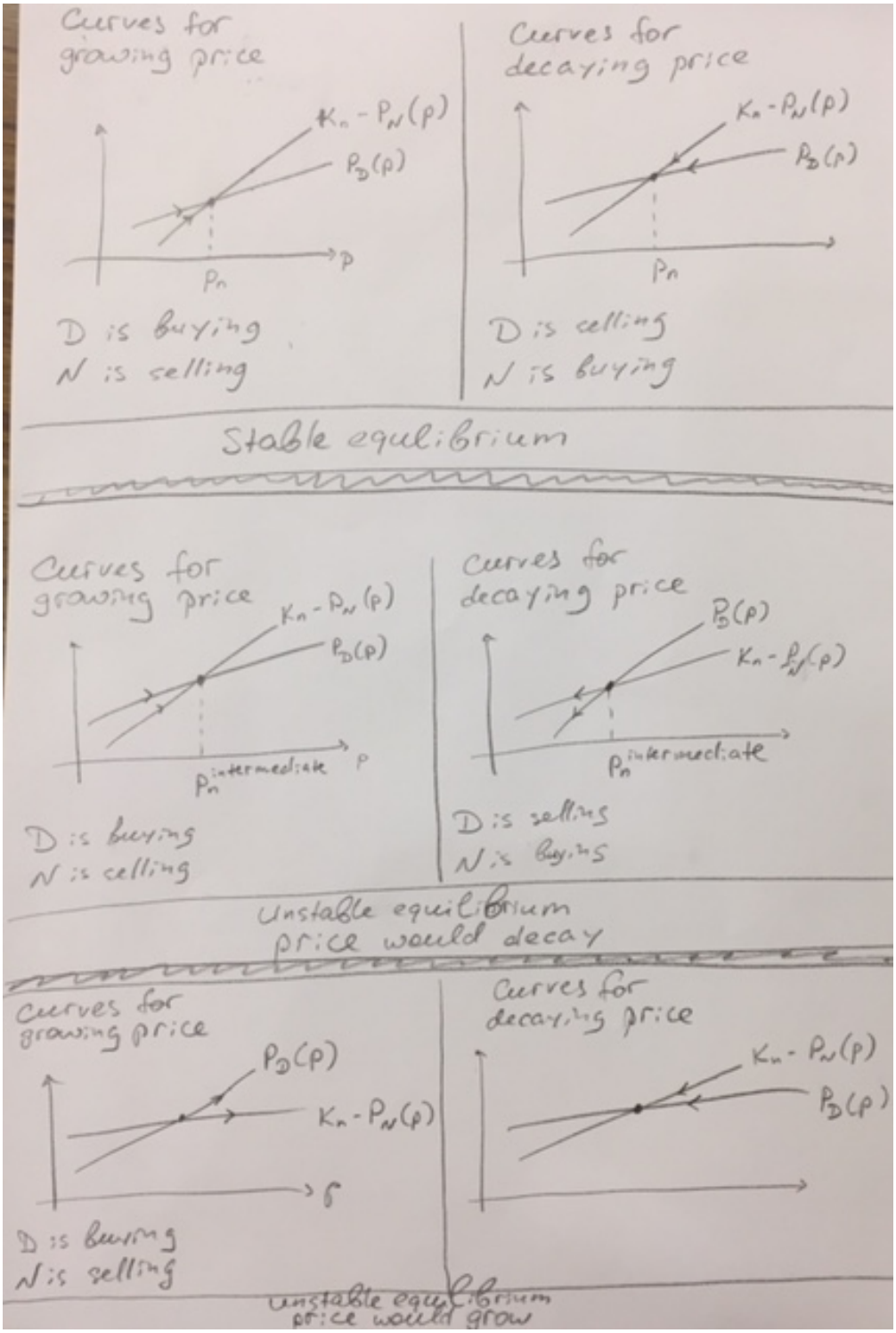


Figure 5.1: Stability of solutions of equation (5.1).

In order to explain how a stabilized price can change and stabilize to the next value, we make the next assumption.

Assumption 5.4 (Transitions between stabilized prices)**.** *Suppose p_{n-1} is a stabilized price at time $n-1$. In particular, it is a stable solution of the equation*

$$P_D(p_{n-1}) + P_N(p_{n-1}) = K_{n-1}. \tag{5.2}$$

We assume that between time moments $n-1$ and n, some external agents E buy or sell some stocks. As a result, the total amount of stocks that is in possession of agents D and N becomes K_n. Moreover, we assume that external agents do not stay at the stock exchange, i.e., the operators (their densities) $P_D(p)$ and $P_N(p)$ do not change in time.

If $K_n \le K_{n-1}$, then external agents E buy $|K_n - K_{n-1}|$ stocks, which increases the price. As a result, agents D also buy. All the stocks bought by agents E and D are sold by agents N. This leads to the increase of the price to a new value p_n^1, where p_n^1 is the smallest solution of the equation

$$P_D(p_n^1) + P_N(p_n^1) = K_n \tag{5.3}$$

satisfying the inequality

$$p_n^1 \ge p_{n-1}. \tag{5.4}$$

Note that small values $|K_n - K_{n-1}|$ may correspond to small changes of the price as in Fig. 5.2 or large changes as in Fig. 5.3. In the latter case, the price jumps up due to the fold bifurcation.

If p_n^1 is a stable solution of equation (5.3)*, then, by definition, it coincides with the new stabilized price:*

$$p_n := p_n^1. \tag{5.5}$$

*Otherwise, the price makes several jumps taking semi-stable values $p_n^2, \dots, p_n^{m-1}$ and a stable value p_n^m (***hopefully with a finite** *m) such that*

$$p_n^2 < p_n^1, \quad p_n^3 > p_n^2, \quad p_n^4 < p_n^3, \quad p_n^5 > p_n^4, \dots \tag{5.6}$$

By definition, we set

$$p_n := p_n^m. \tag{5.7}$$

Analogously, the price will leave the stable value p_{n-1} if $K_n > K_{n-1}$. In this case external agents E sell $K_n - K_{n-1}$ stocks, which decreases the price. As a result, agents D also sell. All the stocks sold by agents E and D are bought by agents N.

The last assumption concerns the strategy of external agents E.

Assumption 5.5. *K_n is a Markov chain. For example, $K_n \sim \mathcal{N}(K_{n-1} + \mu, \tau^{-1})$ with some mean μ and precision $\tau > 0$.*

Remark 5.2. Set

$$G(p) := P_D(p) + P_N(p). \tag{5.8}$$

Then, formally, the relationship between the price p_n and the noise K_n is the same as in Dima's model, cf. (5.1) and (2.3)! Though in our second model, there is a number of further restrictions on admissible values of p_n due to Assumption 5.4.

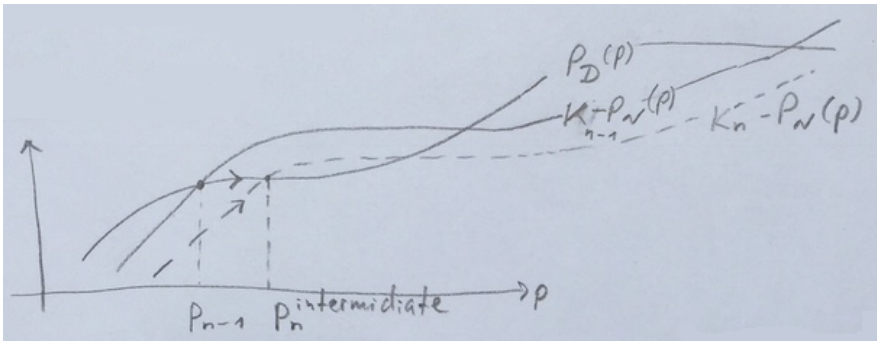
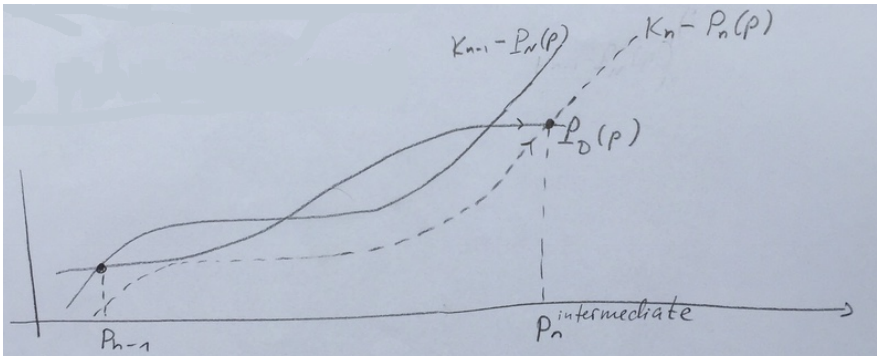


Figure 5.2: Price p_{n-1} gets destabilized and increases without bifurcation.



分支

Figure 5.3: Price p_{n-1} gets destabilized and increases with fold bifurcation.

References

[1] Dima's paper.

[2] J. M. Hernández-Lobato, R. P. Adams. Probabilistic backpropagation for scalable learning of bayesian neural networks. arXiv:1502.05336 [stat.ML] (2015).

[3] C. Bishop. Pattern Recognition and Machine Learning. Springer, 2006.

[4] S. McCarthy, D. Rachinskii, JOP.

[5] Martin Brokate, On the Moving Preisach Model, Mathematical Methods in the Applied Sciences, Vol. 15, 145-157 (1992).