Next-Gen Model Risk Management

High-Fidelity Document Intelligence

Transforming PDF Analysis with AI-Powered Extraction

Alexander Tsoskounoglou

Associate, Model Risk Management — Interest Rate Derivatives (MRM IRD)
Morgan Stanley, Budapest

February 2026

Who Am I

Alexander Tsoskounoglou

- **BSc Computer Science** — solid foundation in software engineering & algorithms
- **MSc Computational Finance** (University of Amsterdam) — quantitative modelling, stochastic calculus & numerical methods
- **Quantitative Analyst** (1 year, hedge fund) — developed & deployed neural-network-based trading strategies in production
- **Quantitative Researcher** (1 year, University of Amsterdam) — led two research projects on Deep Reinforcement Learning for hedging & pricing exotic derivatives

The Current Friction: A Wall of PDFs

The MRM Document Landscape

- **Thousands** of model documents in the Model Context System
- Bermudian Swaptions, SABR Calibrations, CVA/XVA frameworks...
- Dense with:
  - Multi-line integral equations & SDEs
  - Stochastic calculus (Itô, martingales)
  - Nested tables of Greeks & sensitivities
  - **Figures, charts & diagrams —** calibration surfaces, payoff diagrams

Why "Copy-Paste into an LLM" Fails

1. **PDF $\neq$ Text** — PDFs store glyphs, not semantics
2. **Math is destroyed** — integrals $\rightarrow$ dashes, Greek $\rightarrow$ ASCII
3. **Tables collapse** — columns merge into gibberish
4. **Figures vanish** — images are silently dropped
5. **LLM hallucination** — garbage in $\Rightarrow$ *confident* garbage out

⚠ This is not a minor inconvenience.
It is a **model risk**.

The Technical Gap: "If the LLM Can't Read the Math..."

Source: *Markov Functional Market Model* (Oxford, 2008) — HJM Drift Condition & Bond Pricing

× Standard Text Extraction

```
P (t, T ) = EQ exp rs ds ???| t
      - t F

  - T
alpha(t, T ) = sigma(t, T ) · t
t sigma(t, s) ds ????  0 <= t <= T.
P (t, T ) = exp A(t, T ) B(t, T )rt
At betaB + [1][= 0][,]
    - 2 [delta][(][t][)][B][2]
```

- × Integrals → dashes, measures → gone
- × Greek letters → English words
- × Square brackets are OCR artifacts
- × Completely unusable for validation

✓ High-Fidelity Pipeline (Marker/Surya)

$$P(t,T) = \mathbb{E}_{\mathbb{Q}}\left[\exp\left(-\int_t^T r_s\,ds\right)\middle|\mathcal{F}_t\right]$$

$$\alpha(t,T) = \sigma(t,T)\cdot\int_t^T \sigma(t,s)\,ds \;\; \forall\, 0 \le t \le T$$

$$P(t,T) = \exp\big(A(t,T) - B(t,T)\,r_t\big)$$

- ✓ Integral notation & bounds preserved
- ✓ Measure $\mathbb{Q}$, filtration $\mathcal{F}_t$ correct
- ✓ Greek letters $\alpha, \beta, \gamma, \delta$ intact
- ✓ Machine-readable & LLM-ready

Real-World Failure: Longstaff-Schwartz Continuation Value

Source: Same document — Longstaff-Schwartz algorithm (Eq. 4.1).

× Standard Extraction

```
C(w; tk) = EQ[????  j=k+1 N
exp(- ??  rj) I(w, tj; tk, T)] ???Ftk
```

LLM Interpretation:

× Summation sign → "????"
× Integral replaced by "– ??"
× Conditional $\mathcal{F}_{t_k}$ → "???Ftk"

An LLM will hallucinate the missing structure.

✓ High-Fidelity Output

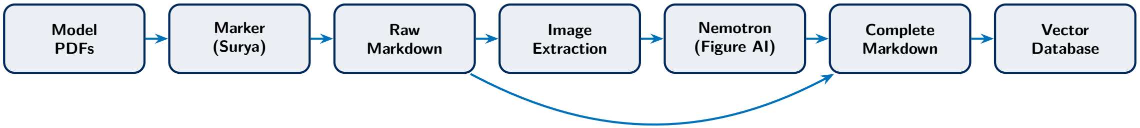
Continuation value at exercise time t_k:

$$C(\omega; t_k) = \mathbb{E}_{\mathbb{Q}}\left[\sum_{j=k+1}^{K} \exp\left(-\int_{t_k}^{t_j} r(s)\, ds\right) I(\omega, t_j; t_k, T)\right.$$

✓ Summation, integral, conditional correct
✓ Validator can cross-reference with code

The equation we use for Bermudan swaption validation.

Proposed Solution: End-to-End Document Intelligence



How It Works

- **Marker:** Converts PDF to Markdown (Surya OCR handles layout/math).
- **Multimodal LLM:** (e.g. Nemotron) reads charts/plots & embeds descriptions.
- **Vector DB:** Stores equation-aware chunks (never split mid-formula).

The End-State Vision

- **Conversational Querying:** Expert answers with equation references.
- **Multi-Doc Comparison:** Cross-reference model specs vs. validation reports.
- **Accelerated Workflows:** Clean context for testing & learning.
- **Code + Paper Reasoning:** Unified queries for code & docs.

Research Summary: Key Technologies

Tool	Role	Strength	Hardware	Key Detail
Marker (+ Surya)	**PDF → Markdown** (core pipeline)	✓ Best-in-class: 95.7 score	GPU or CPU	25 pages/sec on GPU; Surya is the OCR engine.
Docling (IBM)	**PDF → Markdown** (alternative)	✓ Strong table extraction; MIT license	CPU or GPU	Good fallback for table-heavy documents.
Nemotron (NVIDIA)	**1. Figure/Image AI** **2. Quant Risk Chat**	✓ **1M context window**; SOTA open-model	Cost-effective GPUs (A10)	Process entire PVRMs in one pass. Ideal for both visual reasoning & risk analysis conversations.

Strategy

Marker for fast document conversion on-prem. **Nemotron** for deep reasoning and figure understanding.

Why This Matters: Business Value

Speed of Validation

- **Current:** reading 50–400 page PVRM, manually locating equations — *days/weeks*
- **Proposed:** natural language query in seconds.

Task	Current	Proposed
Locate equation	15–30 min	<1 min
Cross-ref docs	2–4 hrs	5–10 min
Review documentation	1–2 weeks	$\sim$3–5 days
Write new tests	Weeks	Days

Broader Impact

- **Reduce model risk:** automated consistency checks between spec and code
- **Replicate & extend tests:** brainstorm new validation tests from the math
- **Assist in Extending & Merging PVRMs:** Help analysts identify shared assumptions/gaps
- **Onboarding:** new analysts query models immediately
- **Audit trail:** logged queries for compliance

Implementation Phases

Phase 1: Proof of Concept

- ✓ Convert `IR_Swaption_Bermudan` PVRM (12 PDFs)
- ✓ Validate math fidelity
- ✓ LLM Context (Copilot)
- ✓ Collect **team feedback**
- ✓ Set accuracy & performance standards

≈ 4–8 weeks

Phase 2: Pilot Vector DB

- ▶ Build vector DB for pilot team
- ▶ Nemotron for Figure AI
- ▶ Iterate on infrastructure
- ▶ Measure productivity gains
- ▶ Infra support needed

≈ 3–5 months

Phase 3: MRM Rollout

- ▶ Deploy across MRM
- ▶ Multi-PVRM cross-ref
- ▶ Analyst Web Interface
- ▶ Training & Docs
- ▶ Model updates

≈ 6–12 months

Month 1–2 → **Month 3–7** → **Month 8–14+**

Technical Architecture

1. Ingestion Pipeline

- **PDF Ingestion:** Process via **Marker (Surya)**
- **Raw Markdown:** High-fidelity math extraction
- **Multimodal Enhancement:**
  - Text & math preserved directly
  - Images processed by **Nemotron**
- **Result:** Complete Markdown (Text + Images)

2. Dual-Store Storage

- **Vector DB:** Equation-aware chunks for ranking
- **Doc Store:** Faithful Markdown for LLM context

3. Query & Retrieval Flow

- **Retriever:** Fetches ranked chunks + full context
- **LLM Engine:** Nemotron generates answers
- **Analyst Interface:** Secure user access

Security & Compliance

- **100% On-Premise:** No external API calls
- **Open Source Stack:** Fully auditable code
- **Access Control:** Respects team permissions
- **Audit Trails:** All queries are logged

Next Steps & Requirements

Phase 1 Actions

1. **Approval** to build local prototype
2. Convert **PVRM IR_Swaption_Bermudan** (12 PDFs)
3. Validate fidelity against original documents
4. Define math fidelity score & criteria
5. Pilot AI assistants & collect feedback

Resource Requirements

Resource	Specification
GPU	A10 or A100 (recommended)
CPU Fallback	Possible (slower)
Storage	~50 GB
Time	4–8 weeks (Phase 1)
Team	1 engineer

The Ask

Approval to allocate compute access and begin Phase 1 prototype on the Bermudan swaption PVRM.

Thank You

Questions?

alexander.tsoskounoglou@morganstanley.com

MRM IRD — Budapest

"The quality of AI output is bounded by the quality of its input."

Appendix: Under the Hood

Equation: Longstaff-Schwartz Continuation Value (Eq. 4.1)

1. Original PDF

$$C(\omega; t_k) = \mathbb{E}_{\mathbb{Q}} \left[\sum_{j=k+1}^{K} \exp\left(- \int_{t_k}^{t_j} r(\omega, s)\, ds \right) I(\omega, t_j; t_k, T) \;\middle|\; \mathcal{F}_{t_k} \right]$$

2. Standard Text Extraction (`pymupdf4llm`)

```
C(w; tk) = EQ[????  j=k+1 N exp(- ??  rj) I(w, tj; tk, T)] ???Ftk
```

3. High-Fidelity Pipeline (`Marker/Surya`)

```
$$C(w; t_k) = \mathbb{E}_{\mathbb{Q}} \Big[ \sum_{j=k+1}^{K} exp \Big( -
\int_{t_k}^{t_j} r(w, s) \ ds \Big) I(w, t_j; t_k, T) ...$$
```

Appendix: Marker Benchmark Performance

Method	Time (sec)	Heuristic Score	LLM Score
Marker	**2.84**	**95.67**	**4.24**
Docling	3.70	86.71	3.70
Mathpix	6.36	86.43	4.16
LlamaParse	23.35	84.24	3.98

Benchmark data from Marker GitHub repository.

Throughput

- Single-page serial: 2.84 s
- Batch mode on H100: **25 pages/sec**
- Can run on **A10, V100, MPS, or CPU**

By Document Type (Heuristic)

Scientific paper	96.67
Financial document	95.37
Legal document	96.69

Financial documents score 95.4/100.

Appendix: Cost & Productivity Estimation

Cost per Query

Platform: Microsoft Azure (MS strategic partner)

Model	In/1M	Out/1M
Nemotron-3-Nano (open-source)	$0.05	$0.20
Gemini 3 Pro (closed, SOTA)	$2.00	$12.00

Typical query: ∼4k input + ∼1.5k output tokens

- Nemotron-3-Nano: **$0.0005**/query
- Gemini 3 Pro: **$0.026**/query

Monthly Cost Projection (per analyst)

Est. queries: 25–40/day × 22 days/month

Scenario	Q/day	Nano	Gemini 3
Conservative	25	$0.28	$14.30
Normal	35	$0.39	$20.02
Heavy use	50	$0.55	$28.60

Team of 10 (Gemini 3): $143–286/month

Estimated Efficiency Gains per PVRM Revalidation

Task	Current	Conservative	Normal	Optimistic
Read & digest PVRM docs	∼16–24 hrs	−10% (**14–22 h**)	−20% (**13–19 h**)	−30% (**11–17 h**)
Locate equations/sections	∼2–4 hrs	−15% (**1.7–3.4 h**)	−25% (**1.5–3 h**)	−40% (**1.2–2.4 h**)
Cross-reference documents	∼4–8 hrs	−15% (**3.4–6.8 h**)	−30% (**2.8–5.6 h**)	−45% (**2.2–4.4 h**)
Interpret figures & tables	∼2–4 hrs	−10% (**1.8–3.6 h**)	−20% (**1.6–3.2 h**)	−35% (**1.3–2.6 h**)
Draft validation tests	∼40–80 hrs	−10% (**36–72 h**)	−20% (**32–64 h**)	−30% (**28–56 h**)
Total per PVRM	**∼64–120 hrs**	**−12%**	**−22%**	**−35%**

Pricing: Azure AI Foundry (Feb 2026). Assumes RAG-based retrieval. Speed-ups assume high-fidelity extraction pipeline is deployed.