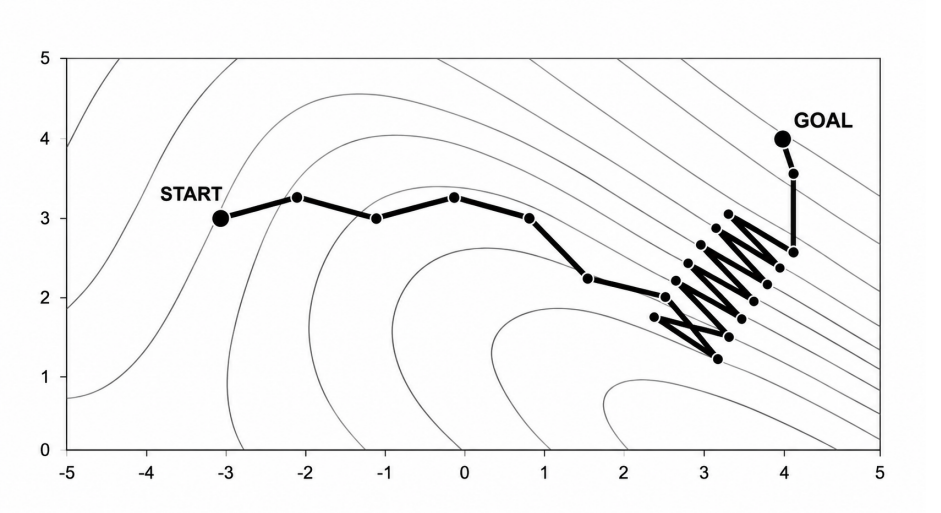
Learn Generative AI Without LLM Complexity

A Comprehensive Introduction Using a Tiny Inference Model (TIM)



Every Decision, Fully Inspectable

Encoding, Learning, and Inference Implemented in Excel, Google Sheets, and HTML/JavaScript

Leonard Brown

Companion applications and resources at `9ParmAI.com`

Learn Generative AI Without LLM Complexity

A Comprehensive Introduction Using a Tiny Inference Model (TIM)

Copyright © 2026 Leonard Brown
All rights reserved.

No part of this publication may be reproduced, distributed, or transmitted in any form or by any means without prior written permission of the copyright holder, except as permitted under applicable copyright law.

Companion Software — Terms of Use

The companion software provided with this book is freely available for educational, research, and personal use, including use by institutions and organizations that charge for their educational services. Users may download, examine, use, and modify the software for these purposes. Commercial exploitation of the software itself, including its sale, licensing, or incorporation into commercial products or services, requires prior written permission from the copyright holder.

All other rights are reserved.

First Edition, 2026
Published by 9ParmAI.com Publishing
Washington, DC, USA
Website: `https://9parmai.com`

Paperback ISBN: 979-8-9822954-9-1
Library of Congress Control Number (LCCN): **pending**

Foreword

The Forest Before the Trees

Much of the discussion of generative AI begins with large language models and then proceeds into the mechanisms from which they are constructed: neural networks, transformers, attention, embeddings, logits, softmax, and related techniques. These mechanisms are important for understanding how an LLM works, but they should not be confused with the more general question of what constitutes a generative AI system.

An analogy may be useful. Generative AI can be thought of as a forest containing many different types of trees. An LLM is one particular type of tree within that forest. Neural networks, transformers, attention mechanisms, and similar technologies describe important features of that tree. Examining them can provide an extraordinarily detailed understanding of LLMs, but studying one type of tree—even in great detail—does not by itself provide a complete picture of the forest.

This book therefore begins with the forest rather than with a particular tree. It first attempts to identify the general elements of a generative AI system without requiring any particular implementation. Only after establishing this broader framework does it turn to a specific system.

The system selected for detailed examination is TIM. TIM is not intended to be a simplified LLM. It is deliberately a simple generative AI system whose principal components and operations can be observed directly.

There is a useful analogy with biology. One reason for studying a simple organism such as an amoeba is not that an amoeba is a miniature human

being, but that a relatively simple organism allows fundamental biological processes to be examined without first confronting the enormous complexity of a human organism. The simpler specimen can make the underlying principles easier to see.

TIM serves a similar purpose. Its simplicity permits the reader to observe domain characterization, prompting, generation, incorporation of generated results, learning, goal seeking, stopping, and failure without first having to master the enormous implementation complexity of an LLM.

The progression of this book is therefore deliberate:

The Forest $\longrightarrow$ A Simple Tree: TIM $\longrightarrow$ A Much More Complex Tree: LLMs

The purpose of TIM is not to reproduce the machinery of an LLM. It is to make the more general machinery of generative AI visible.

This distinction also permits broader questions to be examined without making them dependent upon the particular architecture of an LLM. Later in the book, this framework is used to consider issues including repurposing and dual use, as well as the relationship among functional composition, capability development, and recursive self-improvement.

A second book will build on this foundation. It will assume that the reader understands the general framework developed here and will examine, in detail, a different kind of "tree": generative AI systems whose defining mechanisms include the features commonly associated with large language models, especially neural networks and the architectures built from them.

That second book will not treat an LLM as though it were merely a larger version of TIM. Instead, it will use the framework established in this book to show how the general elements of generative AI are realized within a much more complex implementation. Neural networks, transformers, attention mechanisms, embeddings, logits, softmax, training, and related processes will then be examined as the defining features of that particular class of systems.

The relationship between the two books is therefore analogous to the relationship between introductory and advanced biological study. The first book

examines general properties of generative AI through a relatively simple system. The second presumes that foundation and uses it to investigate more complex systems whose internal mechanisms are substantially different.

The progression across the two books is:

The Forest $\longrightarrow$ A Simple Tree: TIM

$\longrightarrow$ A Complex Tree: Neural-Network-Based Systems

The first book explains the forest and studies a simple tree so that the general structure can be seen. The second book assumes that understanding and examines, in detail, one of the most important and complex types of tree within that forest.

AI Disclosure Statement

A Note on the Use of AI in Producing This Book

Artificial intelligence is both the subject of this book and one of the tools used in producing it. It therefore seems appropriate to describe its role directly.

Three general-purpose AI systems were used in developing this book. ChatGPT, through a paid subscription, was used most extensively. It assisted with model development, technical discussion, research, drafting and revision, testing explanations, and exploring questions arising during TIM's development. Claude, also through a paid subscription, was used primarily for proofreading and independent review. Gemini, generally through its free version, was used occasionally to examine specific questions and provide alternative perspectives.

All three systems were useful, although they served different purposes.

Their contributions went well beyond proofreading. They significantly increased my productivity, helped me examine difficult questions from different perspectives, and supplied background knowledge in areas of AI with which I initially had little experience. Just as importantly, they provided an accessible intellectual sounding board: ideas could be proposed, challenged, reformulated, tested, and reconsidered almost immediately.

This does not mean that the book was "written by AI," a phrase sometimes applied reflexively whenever AI contributes to a work. Such labels can be misleading because they risk discounting serious intellectual effort based on

the tools used rather than the quality, originality, reasoning, and usefulness of the work. The appropriate question is not simply whether AI was used, but how it was used and what responsibility the author exercised in directing, evaluating, verifying, and shaping the result.

The ideas advanced, the book's architecture, the development of TIM, the examples selected, the arguments made, and the judgments about what appeared in the book remained the author's responsibility. AI-generated material was evaluated, modified, rejected, or incorporated, just as information from other tools and sources must ultimately be judged by the person using them.

At the same time, describing AI as merely an editorial convenience would understate its role. Producing this book single-handedly, without AI, would likely have been a multi-year project. With AI as a tool, it was completed in approximately six months.

Humans have always used machines to extend what an individual can accomplish. A calculator extends our ability to perform arithmetic. A spreadsheet makes complex numerical analysis manageable. A word processor greatly expands our ability to draft and revise text. AI extends this progression into a much broader range of intellectual activity, assisting with research, explanation, analysis, questioning, programming, drafting, and revision while drawing upon a vast body of accumulated knowledge.

That does not eliminate the need for human judgment. If anything, it makes judgment more important. AI systems can misunderstand questions, provide incomplete information, make factual errors, or confidently suggest something wrong. Their output must therefore be questioned and checked rather than simply accepted.

The technology discussed throughout this book is AI as a computational system. Producing the book demonstrated another aspect: AI as a tool. Used critically and responsibly, it can substantially extend what one person can investigate, understand, and produce.

Online Resources

Companion Software and Reference Materials

This book is accompanied by computer implementations and reference materials designed to help readers examine, reproduce, and experiment with the concepts developed throughout the text.

The resources include implementations of the Tiny Inference Model (TIM) in Microsoft Excel, Google Sheets, and HTML/JavaScript, together with additional applications demonstrating how TIM can be repurposed for different inference tasks.

All companion resources are available at

https://9parmai.com

Detailed descriptions of the resources, together with instructions for reproducing the book's principal experiments, are provided in Appendix O.

Education and the AI Ethics Debate

An informed discussion of artificial intelligence's ethical implications, risks, and governance requires an understanding of how the technology works.

The author's views on the importance of education in the AI ethics debate are presented in a separate article, available at

https://9parmai.com/ethics

Contents

Foreword **v**

AI Disclosure Statement **ix**

Online Resources **xi**

1 Introduction **1**

2 What Is a Generative AI System? **15**
2.0.1 Optional Pre-Learning 18
2.0.2 Inference 21

3 Pedagogical Structure of the Book **25**

4 The AI Process: Training and Inference **29**
4.1 OPTIONAL PRE-LEARNING 30
4.2 Inference 31
4.3 The Same General Process at Different Scales 33

5 The Tiny Inference Model **35**
5.1 The Logical Structure of TIM 35
5.2 The TIM Domain 36
5.3 Two Characterizations of the Same Domain 40
5.3.1 LOW RES: A Learned Domain Characterization . . . 40
5.3.2 HIGH RES: A Directly Supplied Domain Characterization 42
5.4 Inference 42
5.5 Why TIM Uses Two Resolution Modes 43
5.6 TIM's Logical Structure 44

6 Encoding **47**

6.1 Alternative Encodings 51
6.2 Encoding in Large Language Models 57
6.2.1 A Large Language Model Tokenization Example . . . 58

7 Learning **63**
7.1 Recursive Least Squares Learning 65
7.2 Prediction 66
7.3 Difference 66
7.4 Parameter Update 66
7.5 Freezing the Parameter Weights 68
7.6 Alternative Learning Techniques and Representations 69

8 Generating the Learned Weights **73**
8.1 Generating the Training Data 74
8.1.1 The True Terrain 74
8.1.2 Generating Noisy Training Observations 76
8.1.3 The Nine-Parameter Representation 80
8.2 Encoding 81
8.3 Recursive Least Squares 82
8.4 Reading the TIM Learning Excel Sheet 90
8.4.1 Training Data 91
8.4.2 Encoding Area 92
8.4.3 Learning and Weight Updates 93
8.4.4 Parameter Uncertainty 93
8.5 Conclusion 94

9 The "Generative" in Generative AI **95**
9.1 Defining Generative AI 97
9.2 What Makes an AI System *Generative*? 97
9.3 The Prompt-Dependent Portion 101
9.3.1 GENERATE INCREMENTAL RESULT (OR FINAL RESULT) 101
9.3.2 SHOULD WE STOP? 102
9.4 The Prompt-Independent Portion 103
9.5 The Generation Loop 103
9.6 Incremental Construction 105
9.7 Goal-Directed Generation 106

9.8 Conclusion: The Generation Loop 107

10 The TIM Inference Prompt 109

11 The HTML/JavaScript Implementation of TIM 115
11.1 The TIM Application 115
11.2 Inference 116
11.3 Training 118
11.4 Visualization 120
11.5 LLM Prompt Generator 122
11.6 Documentation 123
11.7 Using an LLM to Explore the TIM Source Code 124
11.8 Validation of TIM's Output Route 126

12 The Generation Loop in Two Lines of Code 129
12.1 Where Is the Complexity? 131
12.2 Generation and Incorporation 131
12.3 The Forest Concept 132

13 Inside the Generation Loop: How TIM Makes Decisions 133
13.1 Introduction: Looking Inside the Generation Loop 133
13.2 The Example: A Route Generated in 21 Hops 134
13.3 Opening the Choices Inspector 138
13.3.1 Example: Examining Hop 6 139
13.4 Why TIM Sometimes Moves Away from Its Goal 141
13.5 Why Simple Decisions Produce a Complicated Route 143
13.6 The Connection with LLM Token Generation 145
13.7 What the Choices Inspector Reveals 148
13.8 The Generation Loop Revisited 149
13.9 Conclusion 150

14 Generation Loops in LLM-Based Systems 153
14.1 The Token-Generation Loop 154
14.2 The Thought-Generation Loop 155
14.3 Nested Generation Loops 155
14.4 A Simple Example 157
14.4.1 The Mathematical Problem 157
14.4.2 How the Two Generation Loops Produce the Answer 159

14.4.3 The Nested Structure 161
14.5 The Same General Structure 162
14.6 Comparison with TIM 163
14.7 A Forest Concept 164

15 TIM and LLMs: Similarities and Differences 165

16 Church, Functional Composition, and Recursive Self-Improvement171
16.1 Church's Functional Formulation of Computation 171
16.2 Church's Lambda Calculus in Modern Terms 172
16.3 Functional Composition 174
16.4 Recursive Composition 176
16.5 From Recursive Composition to Capability 177
16.6 Self-Improvement Is an Increase in Capability 178
16.7 Recursive Self-Improvement 179
16.8 What Church Does and Does Not Tell Us 180
16.9 A Note on Church and Turing 181
16.10A Forest Concept 182

17 Repurposing and Dual Use 185
17.1 Case Study: Dual Use 188
17.1.1 Greatest Unobstructed View 191
17.1.2 Least Visible Location 193
17.2 What the Experiments Demonstrate 195

18 What has TIM Taught Us? 197

19 Conclusion 211
19.1 From Understanding to Experimentation 212
19.2 The Broader Significance 213
19.3 The Forest and the Trees 213

A Using AI as a Learning Tool 215

B Batch and Recursive Calculation of an Average 217

C Derivation of Ordinary Least Squares for a Straight Line 221

D Derivation of Recursive Least Squares for a Straight Line 229

E A Practical Introduction to Vectors and Matrices 239
E.1 Scalars . . . 240
E.2 Vectors . . . 240
E.3 Matrices . . . 241
E.4 Matrix Dimensions . . . 241
E.5 The Transpose . . . 242
E.6 Vector Addition and Scalar Multiplication . . . 244
E.7 The Dot Product . . . 244
E.8 Matrix Multiplication . . . 245
E.9 The Identity Matrix . . . 247
E.10 The Matrix Inverse . . . 248
E.11 The Pseudoinverse . . . 250
E.12 Matrix Differentiation . . . 252
E.13 The Matrix Tools Needed for Least Squares . . . 253
E.14 Matrix Partitioning . . . 254

F Matrix Derivation of Ordinary and Recursive Least Squares 257

G Recursive Least Squares for a Nine-Parameter Surface 281
G.1 The Complete Recursive Calculation . . . 291
G.2 Initializing the RLS Recursion . . . 296
G.3 Training the Nine-Parameter Tiny Inference Model (TIM) . . . 299
G.4 Generating the Training Locations . . . 300
G.5 Calculating the True Terrain . . . 301
G.6 Adding Measurement Noise . . . 301
G.7 Specification of Measurement Noise . . . 302
G.8 Example Training Observations . . . 303
G.9 Different Training Sets Produce Different Learned Models . . . 304
G.10 Probabilistic Training and Deterministic Trained Models . . . 306
G.11 A Probabilistic Model Can Produce Deterministic Outputs . . . 308
G.12 Derivation of Relative and Absolute Parameter Uncertainty . . . 311

H Inference Using the Tiny Inference Model 319
H.1 The Prompt . . . 320
H.2 Candidate Generation . . . 321
H.3 Constraints . . . 322
H.4 Costs . . . 323

H.5 Candidate Selection 325
H.6 The Generation Loop 326
H.7 Stopping Conditions 327
H.8 What Is and Is Not Learned During Inference 328
H.9 Summary 328

I Expanding a Trained Model Without Retraining from Scratch 331
I.1 Case 1: Additional Training Data 332
I.2 Case 2: Expanding the Domain 333
I.3 Case 3: Increasing the Number of Characterization Parameters 334
I.4 Partitioning the Least-Squares Problem 336
I.5 Using What the Existing Model Has Already Learned 337
I.6 Estimating the Newly Added Parameters 338
I.7 What Must Be Retained 340
I.8 Partitioning and Segregated Information 340
I.9 Conclusion 342

J Independent Replication by General-Purpose LLMs. 343
J.1 Independent LLM Replication Test 343

K TIM JAVASCRIPT CODE 349

L Programming with LAMBDA in Excel and Google Sheets 355
L.1 A LAMBDA Is a Function 356
L.2 Naming a Function 357
L.3 LET: Giving Names to Intermediate Results 358
L.4 The Result Convention 361
L.5 A Function Can Return an Array 363
L.6 Iteration Without Explicit Loops 363
L.7 MAP: Many to Many 364
L.8 REDUCE: Many to One 366
L.9 Functions Can Call Other Functions 367
L.10 LAMBDA as a Programming Model 369
L.11 Excel and Google Sheets 369

M TIM EXCEL LAMBDA CODE 371

N Other Small and Inspectable AI Models **379**
N.1 Related Educational Models 380
N.2 A Pedagogical Progression 381

O Downloadable Companion Resources **383**

Bibliography **397**

About the Author **399**

Glossary **403**

Index **411**

Chapter 1

Introduction

Large language models (LLMs) are extraordinarily complex computational systems. They may contain billions or trillions of learned parameters and perform computations through many layers of mathematical operations. Their scale contributes to their remarkable capabilities, but it also makes many of their basic processes difficult to observe directly and, more importantly, difficult to interpret.

This book takes a different approach. Rather than attempting to understand generative AI by simplifying the enormous machinery of an LLM, it constructs a deliberately small, transparent, interpretable, limited-domain generative AI system in which the underlying processes can be observed directly. The resulting Tiny Inference Model (TIM) operates on a nonlinear three-dimensional terrain and is designed specifically for learning, experimentation, and inspection.

TIM is not intended to reproduce the architecture or scale of an LLM. It provides instead a completely inspectable environment in which encoding, learning, learned weights, domain characterization, prompting, inference, and generation can be examined separately and then related to their much larger counterparts in modern AI systems. TIM is therefore not a miniature LLM. It is a small generative AI system constructed so that its operation can be understood in its entirety.

Pre-Learning and Inference

Generative AI systems can typically be understood as operating in two distinct stages: **pre-learning and inference**.

During **pre-learning**, commonly referred to as training, information is acquired from observations and embodied in learned parameter values, commonly called weights. These weights are subsequently used as part of a characterization of the domain about which inference will be performed. The particular learning methodology may vary enormously among systems. TIM uses Recursive Least Squares (RLS); modern LLMs generally use neural-network learning on vastly larger datasets and representations. In both cases, however, learning produces parameter values that encode information subsequently available during inference.

Inference occurs after this pre-learning and can be understood in terms of three components:

Domain Characterization :	What information is available?
Prompt :	What information is sought, subject to what costs and constraints?
Generation Loop :	How is the information sought progressively generated?

The **domain characterization** provides the information available to the inference process. The **prompt** specifies the information sought and the costs and constraints governing its production. The **generation loop** progressively generates the information sought by producing an incremental result, incorporating that result into the computational state, and repeating the process until the requested result or another stopping condition is reached.

Thus, at its most compact, the inference process developed in this book can be represented as

$$\boxed{\text{Domain Characterization} + \text{Prompt} + \text{Generation Loop} \Longrightarrow \text{Generated Result.}}$$

A closely related functional decomposition has long been used in classical AI planning, which separates a domain description, a problem specification, and a generic planner (McDermott et al., 1998). To the author's knowledge, however, this decomposition has not previously been proposed as a general framework for generative AI systems, including large language models.

An important benefit of this functional separation is that each component of inference can be studied separately. The Domain Characterization can be examined independently of the particular problem being posed; the Prompt can be studied independently as the specification of that problem; and the Generation Loop can be examined as the process by which successive results are generated and incorporated. This makes it possible to ask what each component contributes to the behavior of the system rather than treating inference as a single indivisible process.

The separation also permits controlled substitution. Where the components are compatible, one may be changed while the others are held fixed. A different Prompt may be applied to the same Domain Characterization and Generation Loop, or a different Domain Characterization may be used with the same Prompt and Generation Loop. TIM demonstrates both forms of substitution directly.

Functional separation does not imply independence during execution. In complex systems, particularly multi-loop LLM and agentic systems, the components may interact repeatedly: generated results can modify the Prompt or context for subsequent iterations, and the Generation Loop may itself contain nested or interacting loops. The framework therefore separates the functions performed by the components without requiring those components to operate independently.

This separation is important. Learning and generation perform different functions. Pre-learning constructs information that may subsequently be available to the system; inference uses available information to generate a result. RLS is therefore not what makes TIM generative. As developed

later in the book, the generation process itself can be reduced to the simple recursive structure

Get Result → Incorporate Result → Repeat or Stop?

Understanding the generation loop does not require OLS or RLS, even though understanding the complete TIM system includes understanding how its learned representation was constructed.

The Tiny Inference Model

TIM provides a deliberately simple environment in which these distinctions can be observed.

The domain is a nonlinear three-dimensional terrain described by (x, y, z), where (x, y) identifies a location and z represents terrain height. For the experiment developed in this book, the underlying terrain is generated by the Himmelblau function. TIM's learned representation, however, is deliberately much simpler than that underlying terrain.

During pre-learning, TIM receives sequential noisy observations of the terrain. The location variables (x, y) are encoded into nine predetermined functional dimensions, and Recursive Least Squares is used to estimate the corresponding nine parameter weights. Once pre-learning is complete, those weights are frozen.

The resulting nine-parameter surface is a compact characterization of the terrain. Importantly, it is an **approximation**, not the terrain itself. TIM's encoding deliberately omits functional dimensions present in the underlying Himmelblau function and therefore cannot reproduce the terrain exactly. This makes visible an important distinction that can be obscured in much larger systems: a learned representation of an environment is typically a compressed and incomplete characterization of the underlying environment. It captures some features while necessarily omitting or imperfectly representing others.

Within TIM, the Domain Characterization and Prompt perform functionally distinct roles. More generally, they can be treated as functionally separate

components of inference.

The prompt can therefore be changed while keeping the domain characterization unchanged. Conversely, the domain characterization can be changed while keeping the prompt unchanged.

TIM demonstrates this separation directly. It provides two alternative domain characterizations, and the user can switch between them without changing the prompt or any other component of the inference system.

Two Domain Characterizations

TIM can perform inference using either of two domain characterizations.

In **low-resolution mode**, the domain characterization is the nine-parameter surface produced through pre-learning. Inference therefore obtains its information about terrain from the learned and frozen weights.

In **high-resolution mode**, the learned approximation is bypassed and the exact functional representation of the terrain is supplied directly as the domain characterization. The prompt and generation loop can then operate using this more accurate source of information.

This comparison exposes an important point about generative AI. **Pre-learning is not inherently required for inference.** What inference requires is an available characterization of the relevant domain. Pre-learning is one important means of constructing such a characterization, particularly when the underlying environment is not already known. But when an adequate domain characterization is available directly, inference can proceed without first learning it.

The two versions of TIM can therefore be represented as

Observations → Pre-Learning → Learned Domain Characterization → Inference

and

$$\text{Exact Domain Characterization} \rightarrow \text{Inference}.$$

In both cases, the structure of inference remains

$$\boxed{\text{Domain Characterization} + \text{Prompt} + \text{Generation Loop} \Longrightarrow \text{Generated Result}.}$$

TIM thereby permits the source and resolution of the domain characterization to change while the basic inference problem remains the same.

The Prompt: The Problem to Be Solved

The prompt specifies the problem that TIM is to solve. It defines four things: the goal, the constraints governing possible moves, the costs used to choose among those moves, and the conditions under which inference stops.

Goal

The goal is to travel from a starting point, in this case

$$(-3, 3),$$

to a destination,

$$(4, 4).$$

The domain characterization provides TIM with information about the terrain over which this route must be found.

Constraints

At each step, possible moves or hops are subject to six constraints:

1. **Hop length.** An ordinary move is exactly one unit.

2. **Direction.** There are 24 possible headings, separated by 15°.

3. **Course change.** After the first move, the heading must change by more than 15° from the previous heading.

4. **Memory.** A candidate point cannot be within 0.25 units of any previously visited point.

5. **Domain.** The domain extends from -5 to $+5$ in both x and y. A candidate outside the domain is assigned a terrain value of 10,000.

6. **Final hop.** If the destination is within one unit of the current position, the destination itself becomes a possible move. The final hop may therefore be shorter than one unit.

Costs

Each possible move has two costs:

1. **Uphill cost.** Moving uphill has a cost equal to the increase in terrain height. Moving downhill or remaining level has zero uphill cost.

2. **Distance cost.** This is the remaining distance from the candidate point to the destination.

Because uphill cost and distance cost are measured on different scales, each is normalized to a range from 0 to 1 among the candidates available at that step. This allows the two costs to be combined on a comparable basis.

The decision score is

$$\boxed{\text{Decision Score} = \text{Normalized Uphill Cost} + \text{Normalized Distance Cost}}$$

The candidate with the lowest decision score is selected.

Importantly, this is a *local* decision. TIM does not determine the complete route in advance. At each hop, it evaluates the available candidates, selects

the lowest-cost move, incorporates the resulting position into its current state, and repeats the process.

Stopping Conditions

The generation process stops when either:

1. the destination is reached; or
2. the maximum of 1,000 hops is exceeded.

Thus, the prompt can be summarized as

$$\boxed{\text{Prompt} = \text{Goal} + \text{Constraints} + \text{Costs} + \text{Stopping Conditions}}$$

A System Designed to Be Inspected

Working implementations of TIM are provided in Microsoft Excel, Google Sheets, and HTML/JavaScript. These are alternative implementations of the same conceptual system rather than separate models.

The implementations used in this book serve different pedagogical purposes.

The **Excel implementation** is used primarily to explain learning, particularly the Recursive Least Squares (RLS) process through which TIM's nine parameter weights are generated. The spreadsheet format makes the sequential calculations directly visible and allows the learning process to be examined step by step.

The **HTML/JavaScript implementation** is used primarily to explain inference. It is a single, self-contained HTML file in which the prompt, domain characterization, generation loop, and resulting sequence of states can be examined separately. The source code is deliberately uncompressed, extensively documented, and directly inspectable. The objective is not software optimization or production efficiency, but transparency. A reader can examine the source, identify the three major components of inference— (1) Domain Characterization, (2) Prompt, and (3) Generation Loop—examine

and modify their implementation, execute the system, and observe the consequences.

The HTML/JavaScript implementation also includes an **LLM Prompt Generator**. This generates a natural-language specification of the current TIM inference problem that can be submitted to a general-purpose LLM, allowing the LLM to perform the same inference task independently. The result produced by TIM can therefore be compared directly with the result produced by the LLM. In the tests performed for this book to date, the results have corresponded in every case tested. This empirical correspondence does not imply that the two systems will always produce congruent results. If a reader discovers a divergence, the author would be very interested to learn of it and the conditions under which it occurred.

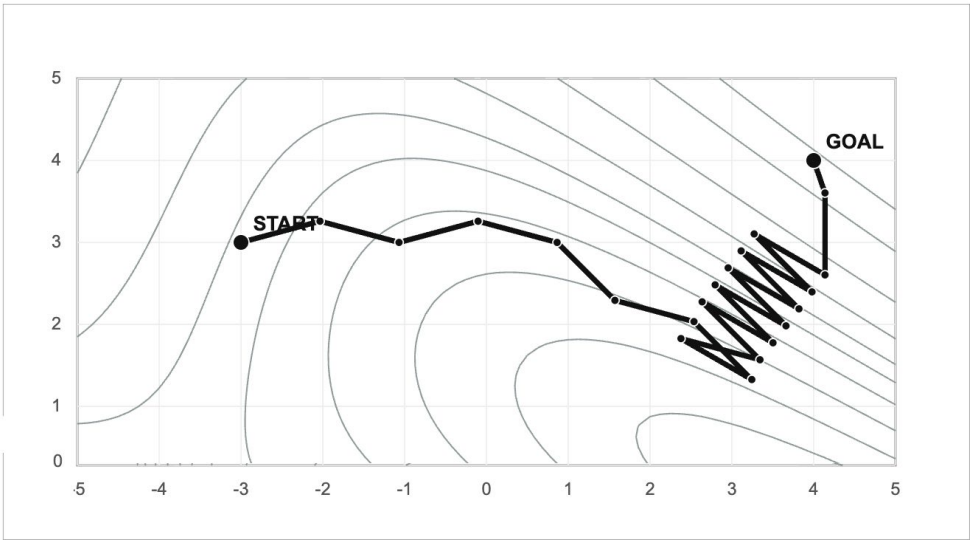


Figure 1.1: TIM Generated Path

The HTML/JavaScript implementation also includes a visualizer that displays the contours of the terrain as TIM sees them, together with the path generated by TIM.

The implementation has separate Learning and Inference sections. Learning exposes the sequential RLS process and the resulting learned domain characterization. Inference separately exposes the prompt, selected domain characterization, generation loop, and resulting sequence of states. The implementation also permits the reader to switch directly between low- and high-resolution domain characterizations.

The structure of the JavaScript itself reflects the conceptual structure developed in the book. The source explicitly separates the prompt-dependent portion of inference, the domain characterization, and the generic generation loop.

This is deliberate. TIM is not merely small enough to understand; **it is deliberately implemented so that it can be understood**.

The Excel and Google Sheets implementations provide alternative spreadsheet-based representations of the same system. They are particularly useful for exposing calculations in familiar rows, columns, cells, and formulas.

From TIM to Large Language Models

The purpose of TIM's simplicity is not to suggest architectural equivalence between a nine-parameter terrain system and a modern LLM. No such equivalence is claimed. Their scale, representations, learning methodologies, and internal architectures differ enormously.

Rather, TIM provides a bridge through which selected computational ideas can first be examined in a system where they are visible.

In TIM, the functional dimensions used for learning are explicitly specified and directly interpretable. Learning determines the weights assigned to those dimensions. In an LLM, representation is vastly larger and substantially learned during training. Text is tokenized, tokens are converted to identifiers, and learned numerical representations are processed through large neural networks. The individual learned dimensions generally do not possess the simple predetermined functional meanings that TIM's dimensions possess.

TIM therefore does not explain an LLM by reproducing one. It provides a small system in which concepts such as encoding, learning, learned weights, domain characterization, prompting, inference, and iterative generation can first be separated and understood.

The reader can then approach their vastly larger counterparts with a conceptual framework already in place.

Representation Is Not Purpose

The separation of domain characterization from inference leads to another important result.

TIM's nine learned parameters characterize the **terrain**, not a particular route through that terrain. They contain information about the domain rather than instructions for one particular use of that information.

Consequently, the same frozen domain characterization can support substantially different inference problems without retraining. A different prompt may seek different information from the same representation, and a different inference procedure may be constructed to extract it. The original TIM route-finding procedure is therefore only one possible use of what has been learned.

The HTML implementation makes this separation especially concrete. Its LLM Prompt Generator can express the current TIM inference configuration as a natural-language specification that can be supplied to a general-purpose LLM, permitting the same numerical task to be executed independently outside TIM.

The domain characterization and the procedure that interrogates it are therefore conceptually distinct.

Repurposing and Dual Use

That distinction leads naturally to the problem of repurposing and dual use.

A case study in this book demonstrates that the same frozen terrain representation can be used to identify locations providing maximum unobstructed visibility, even though TIM was neither trained nor originally designed for that purpose. No new terrain need be learned. The existing domain characterization contains information that can be interrogated for a different objective.

This illustrates a broader point. **The possible uses of information embodied in a learned representation are not necessarily limited to the**

purpose for which that representation was originally constructed.

The underlying information may remain unchanged while the prompt, inference procedure, purpose, and consequences of its use change. A capability may consequently be evaluated positively or negatively depending upon its purpose and the perspective of the observer. Dual use is therefore not merely a theoretical issue associated with hypothetical future AI systems. It can arise from information and computational capabilities available with current technology.

The Purpose of This Book

The purpose of this book is educational.

Modern generative AI systems are extraordinarily large and complex, but scale and implementation complexity should not automatically be confused with complexity of the underlying computational principles. Some of those principles can be studied much more easily in a system deliberately constructed to expose them.

TIM provides such a system.

It allows the reader to observe how information about an environment can be encoded, how observations can be transformed through pre-learning into learned weights, how those weights can form a domain characterization, how a prompt specifies the information sought, and how a generation loop can progressively construct a result.

It also demonstrates something equally important: these components can be separated.

The domain characterization need not have been produced through pre-learning. The same domain characterization can be interrogated for different purposes. The same prompt and generation procedure can operate against domain characterizations of different resolutions. And the generation loop can be studied independently of the particular learning methodology that produced the information upon which it operates.

At the center of the book is therefore a deliberately simple framework:

Domain Characterization :	What information is available?
Prompt :	What information is sought, subject to what costs and constraints?
Generation Loop :	How is the information sought progressively generated?

TIM makes each of these components visible.

That is why it is tiny.