GLY 6739.017S26: Computational Seismology

Notebook 06: My Personal Computing Timeline

Glenn Thompson | Spring 2026

In your week one homework, I asked you to ask senior faculty what computing power they had access to as graduate students.

Here is my personal experience to illustrate how **orders-of-magnitude changes in computing capability** have occurred within a single scientific lifetime — and how those changes altered what scientists had to think about, and what they no longer did.

The BBC Computer Literacy Project and the BBC Micro

In the early 1980s, the BBC launched the **Computer Literacy Project**, an ambitious national initiative designed to ensure that schoolchildren—and the wider public—understood how computers actually worked, not just how to use them. The goal was to demystify computing through television programmes, printed material, and—crucially—a standard educational computer that could be used in schools and homes across the UK.

After evaluating several contenders, the BBC selected **Acorn Computers** to produce a BBC-branded machine. Acorn's design beat proposals from Sinclair, Olivetti, Amstrad, and others because it was fast, robust, expandable, and genuinely programmable, rather than a minimal or toy-like system. The result was the **BBC Micro Model B**.

The BBC Micro was unusually well-suited to education:

- A fast 6502 processor with deterministic performance
- High-quality graphics and sound for its era
- A rich set of I/O ports (serial, parallel, analogue, user port) that encouraged experimentation
- A ROM-based operating system and **BBC BASIC**, so the machine powered on instantly into a programming environment
- Excellent build quality and reliability for classroom use

Most importantly, it was designed to teach **how computers work**, not merely how to load software.

A Personal Starting Point

In **1983**, my dad bought a BBC Micro Model B for our home. Because of the Computer Literacy Project, this was not an unusual purchase: schools were teaching programming, adults attended night classes, and computing was widely understood as a practical literacy rather than a specialist skill.

This was the computer I learned to program on as a child, primarily in **BASIC**, and it shaped how I still think about computers today.

The BBC Micro was not just affordable or popular—it was deliberately engineered to make **programming the default interaction** with a computer.

BBC Microcomputer

Typical specifications:

- **32 kB RAM** (roughly 1 millionth of a modern-day computer)
- **2 MHz CPU**
- **No hard drive**
- Programs loaded from a **cassette tape**
- Only one program could be in memory at once

Despite its limitations, this computer was:

- Fully programmable
- Interactive

- Owned and operated by a single user

In raw instruction speed, it was equivalent to a 1981 IBM PC, but 1/10th of the cost, and less memory.

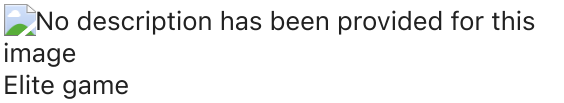
What This Meant Cognitively

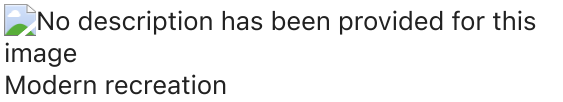
- You were always aware of **memory limits**
- Programs were **small, linear, and explicit**
- You understood exactly what the machine was doing

This environment strongly rewarded **clarity and efficiency**, because waste was immediately visible.

You might think it would be impossible to write decent software, due to the limited memory. I had to write programs in a **highly modular way**, loading only the specific code needed for a task into RAM, then overwriting it with the next routine as the program progressed with the **CHAIN** command. I wrote a spreadsheet application, a word processing, an adventure game, and a soccer management game.

One of the most influential programs I encountered on the BBC Micro was the game **Elite**.


Elite game


Modern recreation

Elite simulated:

- A 3D universe
- Thousands of star systems
- Real-time navigation and combat

All of this ran within **32 kB of RAM**.

Why This Matters Scientifically

Elite is a masterclass in:

- Algorithmic efficiency
- Data compression
- Procedural generation
- Designing within extreme constraints

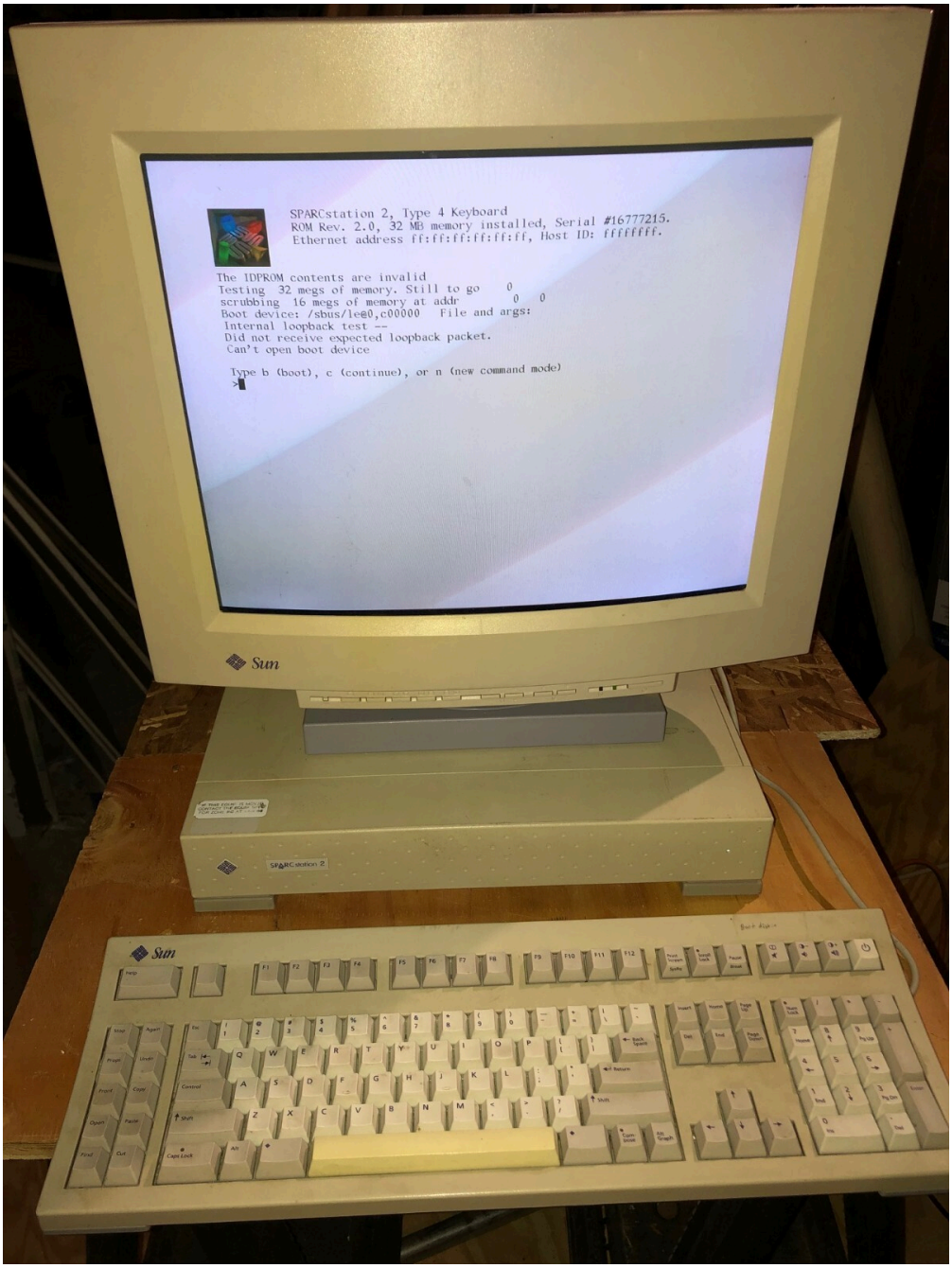
Many early scientists and engineers internalized similar lessons:

The machine will not save you — you must think carefully.

---

2. Scientific Workstations (Mid-1990s)

By **1995**, as a graduate student, I was running synthetic seismogram simulations on a **Sun SPARCstation 2**.



Sun workstation

This system:

- Cost roughly **$15,000**
- Ran **UNIX**
- Was shared across a research group

Compared to the BBC Micro, it was approximately:

- **~1,500× faster** in floating-point computation
- **~1,000× more RAM**
- Equipped with a **real hard drive** and fast disk I/O

What Changed

- Computation became **interactive**
- Visualization became routine
- Batch processing faded from daily use

What Didn't Change

- You still thought carefully about **CPU time**
- Code efficiency still mattered
- Large simulations were still expensive

---

3. A Modern Desktop (Today)

Today, I use a **Mac mini M4** — a consumer-grade machine.

Mac Mini M4

Modern Mac's using Apple Silicon M1, M2, M3, or M4 processors. These are ARM (Acorn RISC machine) processors, which track their lineage back to Acorn Computers and the BBC Micro. They are lower power than Intel x86 processors, and have been used in low

power devices such as smartphones, tablets, and Raspberry Pi computers for over 15 years now.

Orders-of-Magnitude Comparison Across Three Computing Eras

Dimension	BBC Micro Model B (1983)	Sun SPARCstation 2 (1995)	Mac mini M4 (2025) vs SPARCstation 2	Mac mini M4 (2025) vs BBC Micro
Typical user	Individual / student	Research group	Individual scientist	Individual scientist
RAM	**32 kB (and 32 kB ROM)**	**64 MB**	**24 GB**	**24 GB**
RAM increase	—	**1,000×**	**~400×**	**~400,000×**
CPU performance	Baseline	**~1,500× faster (FP)**	**~30,000× faster**	**~45 million × faster**
GPU performance	None	None	**~7 million×**	**~10 billion**
Storage	Cassette tape: 60 kB 5.25″ floppy disk: 100 kB	SCSI hard drive: 200 MB (2000x floppy)	NVMe SSD: 512 GB (2,500x)	NVMe SSD: 512 GB (5 million x floppy)
Disk I/O speed	(Cassette: 66 B/s, sequential: 10-30 minutes load time) Floppy: 12.5 kB/s, random access	SCSI: 2 MB/s (160x floppy)	**6 GB/s (3000x)**	**(500,000x floppy)**
Multitasking	None	True multitasking (UNIX)	True multitasking	True multitasking
Interactivity	Immediate but limited	Interactive computing	Fully interactive, real-time	Fully interactive, real-time
Dominant constraint	**Memory & I/O**	**CPU time**	**Complexity & scale**	**Complexity & abstraction**
Cognitive mode	Explicit, careful, minimal	Efficient, planned	Exploratory, assumption-heavy	Exploratory, abstraction-heavy

Cognitive Consequences

- You rarely think about memory
- You prototype freely

- You expect results *now*
- Inefficiency is often invisible

This changes how scientists work — and how mistakes happen.

---

4. What Disappeared — And What Replaced It

Across this timeline, several concerns have **disappeared**:

Then	Now
Counting kilobytes	Thinking in gigabytes or terabytes
Manual memory management	Garbage collection & abstractions
Batch job submission	Interactive workflows
Scarce CPU time	Abundant compute, expensive insight

But these were replaced by new constraints:

New constraints
Software complexity
Reproducibility
Automation failure modes
Interpretability of models
Long-term sustainability

---

5. Connecting Back to the Course

> **The history of computing is the history of deciding what humans should no longer have to think about.**

In seismology:

- Instruments removed the need to watch drums
- Digitizers removed the need to digitize by hand
- Continuous archives removed the need to decide what to save
- Python removed much of the friction in building workflows
- Machine learning removed explicit rule-writing — but introduced new risks

Understanding modern seismology therefore requires not just knowing *how* to compute, but knowing **which abstractions you are relying on — and when they might fail**.

Reflection Prompt

Consider your own computing experience:

- What is one thing you **never think about**, but scientists once had to?
- What new responsibility has replaced it?
- How might that affect scientific reliability?

Why This Notebook Matters

This notebook is not about nostalgia. It is about **calibration**.

When you work with modern seismic systems — digitizers, automated detectors, machine-learning pipelines — you are operating atop decades of accumulated abstraction. The more powerful the system, the easier it is to forget what lies underneath.

The goal of this course is not to make you slower or more cautious — it is to help you become **deliberate**.