GLY 6739.017S26: Computational Seismology

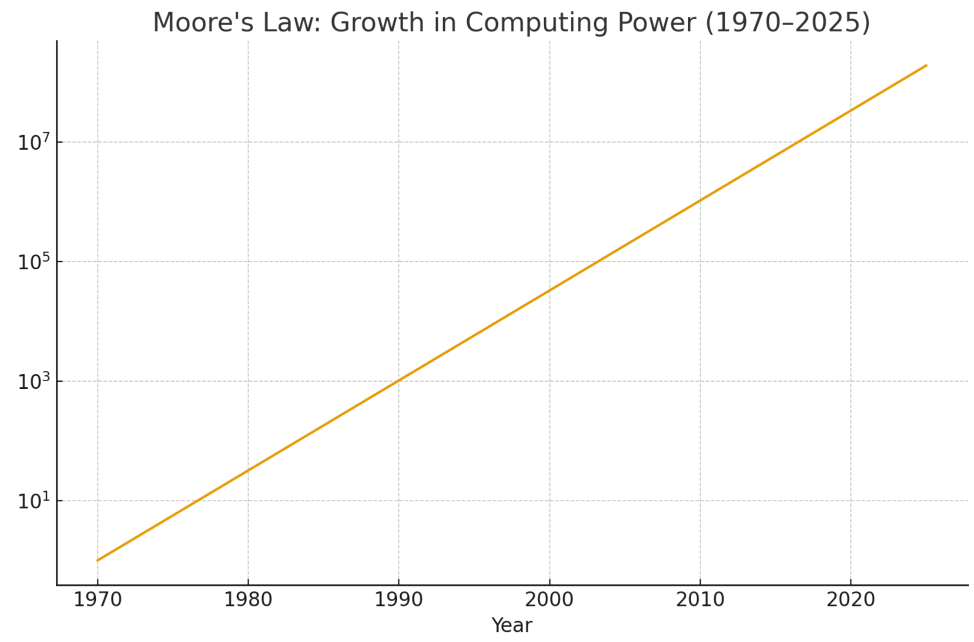
Notebook 04: History of Computing (Part 2) - 1973 onwards

Glenn Thompson | Spring 2026

In part one we looked at the period from 1936-1972. We will now look at 1973 onwards.

Moore's Law

Moore's Law is the observation by Intel co-founder Gordon Moore in 1965 that the number of transistors on a microchip doubles approximately every two years, leading to exponential growth in computing power, smaller devices, and reduced costs



This graph starts in 1970, because that is around the time when computers began to be used in Seismology.

1970s: Centralized Computing, FORTRAN Dominance, and the Birth of UNIX

In the **early 1970s**, scientific computing—including seismology—was almost entirely **centralized**. A graduate student or researcher would typically rely on a **shared institutional mainframe or departmental minicomputer**, rather than any personally owned or laboratory-dedicated machine. These systems ran vendor-specific operating systems and were accessed through terminals or indirectly via batch submission.

Seismological computing at this time was used **primarily for earthquake location and magnitude determination**, not waveform analysis. Seismic signals were recorded on **analog helicorder drum records**, and phase arrival times and amplitudes were **measured manually** using rulers, magnifiers, and timing marks. These measurements were then transcribed into numerical form and entered into computer programs—most commonly written in **FORTRAN**—via **punch cards** or later magnetic tape.

Jobs were submitted to computing centers running systems such as the **IBM System/360**, **CDC** mainframes, or the **DEC PDP-10**, and executed in **batch mode**. Turnaround times ranged from hours to days, making debugging slow and highly iterative. Output was returned as **printed tables or pen-plotter graphics**, rather than interactive screen displays.

Nearly all core seismological algorithms of the era—earthquake location, magnitude calculation, and travel-time inversion—were implemented in FORTRAN IV and later **FORTRAN 77**, reflecting the language’s dominance in numerical scientific computing. Hypocenter solutions were initially computed using early **Geiger-type location methods**, and increasingly with standardized community codes such as **HYPO71**, released in **1971**, which rapidly became the global standard for earthquake location.

In summary, 1970s seismology computing was defined by **centralized hardware**, **batch workflows**, **manual data extraction**, **FORTRAN-based numerical codes**, and the **UNIX operating system**.

1977: Personal (Desktop) Computer Revolution

The first **commercially successful personal (desktop) computers** reached the mass market in **1977**, most notably the **Apple II** and the **Commodore PET**. These machines created the modern home-computer industry and defined what a *personal computer* would become: a **fully programmable system owned and operated by a single individual**, sitting on a desk rather than in a computing center.

Apple 2

Commodore PET

Universities rapidly adopted these early microcomputers for **teaching programming**, introductory numerical methods, word processing, and spreadsheets. While they were not yet powerful enough to replace institutional mainframes or departmental minicomputers for serious scientific workloads, they profoundly changed how students *learned* to compute—shifting programming from a centralized, permission-based activity to an everyday skill practiced interactively.

From a **seismological perspective**, however, this period still belonged firmly to the **mainframe and minicomputer era**. Earthquake location, magnitude calculation, and travel-time modeling continued to run on IBM, CDC, and DEC systems using **FORTRAN** codes derived from Geiger-type algorithms and standardized community programs such as **HYPO71** (released **1971**). Desktop machines were typically used only for **learning, plotting, code development, and report writing**, not for operational data processing.

The formal standardization of **FORTRAN 77 in 1977** was especially important. It became the dominant scientific programming language across both mainframes and emerging desktop systems, ensuring **long-term portability of seismological codes** across computing platforms for decades. Thus, while **Apple II–class machines transformed education and personal productivity**, real-time seismic monitoring and routine earthquake catalogs still depended on **large shared computers**. The personal desktop had arrived—but the scientific infrastructure had not yet followed.

1980s: IBM PCs, the Attack of the Clones, and the Birth of Digital Seismology

The **IBM Personal Computer**, introduced in **1981**, marked a pivotal shift toward standardized personal computing. Built around the **Intel 8088** processor and an openly documented hardware architecture, the IBM PC unintentionally enabled a vast ecosystem of **x86-compatible "PC clones."** By **1982–1983**, companies such as **Compaq** demonstrated fully compatible systems through clean-room BIOS reverse-engineering, triggering rapid proliferation. By the **mid-to-late 1980s**, IBM-compatible PCs had become dominant in business, government, and universities, consolidating software development around **MS-DOS** and, later, early versions of **Microsoft Windows**.

Original IBM PC

IBM-Compatible Clone

In parallel, the **Classic Mac OS** (introduced in **1984**) popularized the graphical user interface in universities for writing, teaching, and graphics. However, Macintosh systems remained largely **separate from scientific UNIX computing** during this decade and played only a minor role in operational seismology.

For seismology and other physical sciences, the decisive computing platform of the 1980s was the **UNIX workstation**—not the PC. Systems from Sun, SGI, and DEC ran **native UNIX**, supported multi-user networking, and provided the performance and I/O capabilities needed for **real-time data acquisition and processing**. Programming was split cleanly: **C** dominated systems and acquisition software, while **FORTRAN 77** remained the primary language for numerical seismology and legacy algorithms.

By the **mid-1980s**, a graduate student in seismology typically worked in a **hybrid computing environment**. Core seismic processing—now increasingly including **automated event detection, phase picking, earthquake location, and magnitude calculation**—ran on **PDP-11 and VAX-class systems** under UNIX or VMS, accessed via text terminals. At the same time, **IBM PCs and early clones** were becoming common on desks for coding, plotting, and document preparation.

Crucially, this decade marked the true transition from **analog to digital seismology**. **Analog-to-digital converters (ADCs)** were installed directly at observatories, enabling **continuous digital waveform streams** to be processed in near-real time for the first time. This transformation was constrained by **severe storage and telemetry**

limitations: disk space was scarce, magnetic tape was expensive and slow, and radio or telephone telemetry bandwidth was limited.

These constraints made **automated STA/LTA (short-term average / long-term average) detectors central to digital seismology**—not merely as alerting tools, but as **essential data-reduction mechanisms**. STA/LTA algorithms ran continuously on streaming telemetry, **detecting events in real time and selectively triggering short waveform windows for permanent storage**. As a result, the entire workflow—**event detection → phase picking → hypocenter location → magnitude calculation**—became, for the first time, a unified **digital signal-processing pipeline**.

Throughout much of the 1980s, **analog drum helicorders continued to operate in parallel** with early digital systems, producing a mixed observational environment in which human analysts visually interpreted paper records while computers increasingly “watched” the Earth automatically. The 1980s therefore represent the decisive moment when seismic monitoring crossed from **manual, observer-driven practice to computer-driven, real-time digital observatories**, setting the stage for fully continuous digital networks and global archives in the 1990s and beyond.

Sidebar: Why MS-DOS & Windows?

Microsoft's first operating system was XENIX, a Unix-based operating system, which they began working on in 1978 and released in 1980 - and thought it would be its future. MS Word was first written for it. So why were MS-DOS and MS-Windows developed at all? It really came down to timing.

The simple answer is that desktop PCs (microcomputers) in 1981 had very little power. And Microsoft were struggling to run Xenix on a 1981 IBM PC.

The Apple 2 and many other microcomputers used CP/M ("Control Program/Monitor"), which was very simple compared to Unix that dominated mini and mainframe computers. Tim Paterson ported this to the 8086 CPU in 1980 and called it "86-DOS" (8086 Disk Operating System). IBM was a hardware company but they needed an operating system for the IBM PC. IBM asked Bill Gates (Microsoft CEO), who was on the IBM board, to provide one. Microsoft acquired the rights to 86-DOS in December 1980 for $75,000, and rebranded it as MS-DOS (Microsoft Disk Operating System). This put Microsoft in pole position to become the world's biggest application software company, because it was now developing the operating system installed by default on most of the world's computers! And it could kill off other software companies like Lotus and Borland, by making Microsoft clones of their spreadsheet and database programs. IBM thought the hardware market was what mattered, but IBM-clones rapidly ate away at their business, while Microsoft overtook them as the dominant computer company in the world. To compete with the MacOS graphical desktop, MS-Windows was released in 1983 on top of MS-DOS.

So what became of XENIX? It was Microsoft's internal system throuhgout the 1980s, used for their development. It was commercially successful, and by the late 1980s more than half of Unix installations globally were XENIX. But because it wasn't quite ready in 1981, MS-DOS and later MS-Windows cornered the market. Xenix was better, it just wasn't reliable on such a weak PC.

1990s: UNIX Everywhere, Linux Emerges, and the Era of Continuous Digital Seismology

The **1990s** marked the full maturation of **desktop computing, global networking, and digital scientific infrastructure**. In the early part of the decade, **UNIX workstations** from vendors such as Sun Microsystems, Silicon Graphics, and DEC were common on scientific desktops, offering strong floating-point performance, advanced graphics, and true multi-user networking. In parallel, **IBM-compatible PCs** based on rapidly advancing **x86 processors** gained hardware floating-point units, larger RAM capacities, and fast local disks, making them increasingly viable for technical computing.

A pivotal development was the emergence of **Linux** in **1991**. Strictly speaking, Linux refers to the **operating-system kernel**; the complete systems adopted by scientists in the 1990s were more accurately **GNU/Linux** distributions, combining the Linux kernel with **GNU compilers, shells, libraries, and core utilities**. In practice, the term *Linux* quickly became the common shorthand for this full UNIX-like environment. By the **mid-to-late 1990s**, Linux running on commodity PCs began displacing proprietary UNIX workstations as the dominant scientific desktop and server platform, dramatically lowering costs while preserving UNIX compatibility.

At the same time, the rapid global expansion of **Ethernet networking and the Internet** enabled routine file transfer, remote logins, and near-real-time data exchange between institutions. For the first time, seismic networks could operate as **distributed, networked systems**, rather than isolated local installations.

Universities developed a clear **two-tier computing culture** during this decade. **Undergraduate students** were typically trained on **IBM PCs** running **Windows 95**, **Windows 98**, or **Windows NT**, or on **classic Mac OS**, for word processing, spreadsheets, graphics, presentations, and basic programming. Email and routine Internet use became standard by the mid-1990s. In contrast, **graduate students in the physical sciences** were expected to become fluent in **UNIX environments**, command-line workflows, and compiled programming languages, including manual compilation, linking, and job control.

Programming practices reflected the hardware constraints of the era. **C** and **FORTRAN 77**, later joined by **FORTRAN 90**, remained the workhorses of numerical modeling and real-time seismic processing. **C++** gained traction for large software systems. Interpreted languages were increasingly used for scripting and automation, with **Perl** becoming especially important for telemetry handling, data parsing, workflow automation, and early web services. **MATLAB** also emerged as a powerful high-level environment for numerical analysis, visualization, and teaching. Despite these advances, **performance-critical seismology remained dominated by compiled code**, as CPU speed and memory bandwidth were still the principal bottlenecks.

For seismology, these computing and networking advances enabled a **fundamental shift from trigger-based digital monitoring to fully continuous digital observatories**. By the early 1990s, the combination of **cheap hard disks**, affordable high-capacity tape systems such as DAT and DLT, and improved telemetry bandwidth made it economically feasible to **archive continuous waveform data at scale**. Seismic networks therefore transitioned from storing only triggered event windows to maintaining **continuous digital archives**, preserving not only earthquakes but also background signals such as volcanic tremor, ambient noise, and long-period deformation that had previously gone unrecorded.

At the same time, **real-time automated processing pipelines matured**. Systems such as **Earthworm** and **Antelope** integrated **continuous acquisition, STA/LTA detection, phase picking, event association, hypocenter location, magnitude calculation, and database archiving** into unified real-time workflows. Detection algorithms were no longer used primarily as storage filters; instead, they became components of **end-to-end digital monitoring systems** feeding live catalogs, alarms, and early web-based displays.

By the end of the decade, most regional and national seismic networks were **continuously digital, networked, and largely automated**. Analysts shifted from watching paper drum helicorders to **interacting with live waveform displays, digital spectrograms, and real-time catalogs on networked UNIX workstations and PCs**. The **1990s thus represent the decade in which seismology completed its transition into a modern, continuously recorded, networked digital science**, laying the technological and cultural foundation for today’s real-time global monitoring systems.

2000s: Linux as the Scientific Default, Mac OS X Joins UNIX, and the Rise of Large-Scale Computational Seismology

The **2000s** completed the transition to **Linux as the default operating system for scientific computing**, spanning individual desktops, departmental servers, clusters, and national supercomputers. By the early part of the decade, inexpensive **x86 PCs running**

Linux had largely displaced the remaining proprietary UNIX workstations as the standard scientific desktop, while preserving full UNIX compatibility at dramatically lower cost.

A pivotal parallel development occurred in **2001**, when Apple introduced **Mac OS X**. Built on a UNIX foundation with a native POSIX shell, Mac OS X transformed Macintosh computers into **first-class UNIX scientific workstations**. For the first time, researchers could use Macs interchangeably with Linux systems for SSH access, compilation, scripting, and open-source scientific software, helping to unify scientific workflows across platforms.

At the same time, universities and research centers increasingly assembled **Linux clusters** from commodity hardware connected by fast Ethernet and, later, early InfiniBand networks. These clusters enabled parallel processing at a fraction of the cost of traditional supercomputers. Rapid growth in disk capacity—driven by inexpensive IDE and later SATA drives—made **multi-year continuous waveform archives routine** rather than exceptional. Storage constraints that had shaped seismic data workflows for decades effectively disappeared.

This decade also saw the consolidation of an **open-source scientific software ecosystem** in seismology. Community-developed packages for **data acquisition, real-time processing, visualization, and archiving** became widely adopted, replacing many locally maintained legacy systems. Seismic networks were now fully **Internet-connected**, enabling near-instant data sharing between observatories, national data centers, and global archives. Continuous waveform data, metadata, and earthquake catalogs increasingly followed standardized formats, laying the groundwork for modern interoperable seismological infrastructure.

Programming practices reflected both continuity and change. **FORTRAN (77/90/95)** remained dominant for legacy modeling and wave-propagation codes, while **C and C++** continued to underpin real-time systems and large processing frameworks. **MATLAB** became ubiquitous in teaching, signal processing, and visualization. **Perl** remained critical for telemetry handling, automation, web backends, and data formatting. At the same time, **emerging Python numerical libraries** began to appear in research workflows, primarily for visualization, post-processing, and experimentation as computing power increased—though performance-critical components were still overwhelmingly implemented in compiled languages.

From a research perspective, the 2000s enabled **computationally intensive waveform modeling and large-scale data analysis** on university desktops and modest clusters. Techniques such as **finite-difference and spectral-element wave propagation modeling, frequency–wavenumber (f–k) analysis, waveform cross-correlation, and network-scale inverse problems** became routine rather than exceptional.

Operationally, real-time seismic monitoring systems matured into **continuous, automated, and web-facing infrastructures**. Analysts increasingly interacted with **live**

waveform browsers, digital spectrograms, and continuously updating event catalogs, while automated detection, association, and location pipelines ran continuously in the background. Human effort shifted away from basic signal discovery toward **interpretation, quality control, and hazard communication**.

By the end of the **2000s**, seismology had fully become a **data-intensive, networked computational science**, characterized by continuous global waveform archives, Linux- and UNIX-based computing platforms, open-source software ecosystems, and routine large-scale numerical modeling. This transformation set the stage for the **Python-driven, machine-learning, cloud-based, and citizen-science revolutions** that would follow in the 2010s.

2010s: Python, Open Science, Platform Convergence, and the IRIS Data Revolution

The **2010s** marked a decisive shift toward **high-level open scientific programming**, enabled by the convergence of **multi-core computing, inexpensive multi-terabyte storage, and pervasive high-speed Internet connectivity**. Memory and disk capacity increased by orders of magnitude, solid-state storage became routine, and academic networks achieved sustained high-throughput global data transfer. Computing was no longer a scarce resource to be carefully rationed, but instead became **standard infrastructure**, transforming seismology into a fully **data-intensive scientific enterprise**.

At the same time, **platform differences largely faded in importance**. Linux, macOS, and cloud systems all presented **UNIX-like environments**, allowing the same tools, workflows, and software stacks to run across laptops, servers, clusters, and national cyberinfrastructure. In practice, a stable division emerged: **Windows** remained dominant for general-purpose education and administration, **Linux** dominated servers, clusters, and large-scale cyberinfrastructure, and **macOS** became a common UNIX research desktop. For researchers, however, the experience of scientific computing became increasingly **platform-agnostic**.

For seismology, the most profound institutional change of the decade was the maturation of the **IRIS Data Management Center** as the **central global hub for seismic waveform data, metadata, and earthquake catalogs**. Prior to this era, observatories and regional networks typically stored data locally, and researchers often had to contact individual institutions directly to request waveform segments or catalogs. During the 2010s—made possible by cheap large-scale storage and fast Internet links—data-producing networks could **continuously stream and archive real-time data to a centralized, professionally managed facility**, while researchers gained **a single global portal** for standardized data discovery and retrieval. This fundamentally shifted seismology from a **distributed, institution-by-institution access model** to a **globally centralized, web-based data ecosystem**.

In parallel, the **real-time monitoring software landscape** continued to evolve. Long-standing systems such as **Earthworm** and **Antelope**, both originating in the 1990s, remained widely used in operational networks but diverged in emphasis. Earthworm, developed at the USGS as a fast, modular real-time detection system, increasingly incorporated database-driven catalog production and archival capabilities. Antelope, developed commercially by BRTT, retained its strength as a tightly integrated **waveform–metadata–catalog–database platform** for large national and regional networks. During the same period, the open-source **SeisComP** system matured rapidly and was widely adopted as a modular, end-to-end solution for real-time acquisition, processing, association, location, and alerting.

Equally transformative for research practice was the emergence and rapid adoption of **ObsPy**. Built on **Python**, ObsPy standardized waveform I/O, metadata handling, catalog access, signal processing, detection, and visualization within a single, extensible, object-oriented framework. For the first time, researchers could **rapidly prototype complete seismic processing pipelines**—from data discovery and download through filtering, measurement, and visualization—using high-level scripted workflows rather than large monolithic compiled codes. Python quickly became the **dominant language for research workflows, pipeline development, and reproducible analysis**, while legacy FORTRAN and C/C++ codes increasingly served as performance-critical back-end engines wrapped by Python interfaces.

The net effect of these changes was a profound shift in research **scale and style**. Instead of working with small, locally curated datasets, investigators could now routinely analyze **continental-to-global waveform archives spanning thousands of stations and decades of time**. Studies of **ambient noise tomography, global wavefield imaging, network-wide tremor detection, and large-scale statistical earthquake behavior** became standard research practice rather than exceptional efforts.

By the end of the decade, seismology had entered a new operational regime in which **computing power, storage capacity, and network bandwidth were no longer the primary limiting factors**. Instead, the dominant challenges increasingly became **data volume, algorithmic scalability, software sustainability, and reproducibility**, marking the full transition of the field into the modern **open, big-data era of Earth science**.

Comparison Box: Earthworm vs Antelope vs SeisComP (Operational Seismology, 1990s–2010s)

System	Origin & Era	Core Strengths	Typical Deployment	Key Limitations
Earthworm	Developed at USGS in the **1990s**	Fast real-time detection, modular ring-based architecture, low-latency triggering	Small to medium regional networks, volcano observatories, research testbeds	Historically weak database integration; catalog and archive tools added later to compete with Antelope
Antelope	Developed by BRTT in the **1990s** as a commercial system	Fully integrated **waveform–metadata–catalog–database** architecture, strong archival and QC tools	Large national and regional networks, long-term operational archives	Proprietary licensing; higher cost and administrative overhead
SeisComP	Open-source system emerging strongly in the **2010s**	End-to-end acquisition, detection, association, location, alerting; highly modular; native FDSN services	National networks, regional observatories, cloud and distributed deployments	Newer ecosystem during early adoption; migration effort from legacy systems

Key institutional trend of the 2010s:
Earthworm expanded toward **database-driven catalog production** to compete with Antelope's long-standing integrated data model, while **SeisComP emerged as a modern, open-source alternative** capable of replacing both legacy proprietary and research-focused systems for fully integrated real-time network operations.

2020s: Cloud Computing, ARM Processors, AI/ML, and the Era of Continuous Global Analysis

The **2020s** have been defined by the convergence of **cloud computing, GPU acceleration, machine learning, and energy-efficient ARM-based processors**, building on the already mature foundations of high-speed networking and low-cost massive storage. Scientific computing is no longer tied primarily to local desktops or even institutional clusters. Instead, **elastic, on-demand cloud resources** allow researchers to scale computation dynamically—from a single laptop to thousands of distributed cores—while paying only for what they use. At the same time, **GPUs** have become standard for large numerical and learning-based workloads, and **ARM processors**, long dominant in mobile devices, have entered high-performance laptops, desktops, and servers due to their exceptional performance per watt. Solid-state storage, object storage, and multi-petabyte archives are now routine components of research infrastructure.

For seismology, these hardware and infrastructure trends have enabled a shift from **large-scale data access to large-scale computation on demand**. Continuous global waveform archives—centralized through infrastructures such as the IRIS Data Management Center and its successors—can now be processed **at planetary scale**. Analyses spanning **decades of continuous data from tens of thousands of stations** have become routine rather than exceptional. Tasks that were once confined to small regions, such as dense waveform cross-correlation, ambient noise tomography, or network-wide detection, can now be executed globally using cloud-based or GPU-accelerated workflows.

The most visible methodological shift of the decade has been the **rise of machine learning and artificial intelligence**. Neural-network–based phase pickers, detectors, classifiers, and event associators now operate at **continental to global scale**, often exceeding traditional rule-based algorithms in sensitivity and consistency. These approaches depend directly on the data and infrastructure revolutions of the 2010s: without centralized archives, fast networks, and massive storage, **training data-hungry ML models would not be feasible**. As a result, the primary computational bottlenecks have shifted away from data access and toward **model training cost, validation, interpretability, bias, and reproducibility**.

At the same time, **real-time processing has expanded outward toward the network edge**. Affordable, low-power multi-core systems—often ARM-based—now support local buffering, preprocessing, and intelligent triggering near sensors, while cloud infrastructure handles aggregation, large-scale processing, and long-term archiving. Seismological computing increasingly operates as a **distributed continuum**, spanning edge devices, observatories, institutional clusters, and global cloud platforms.

Programming practices reflect this shift toward scalability and human efficiency. Modern seismology operates across a **heterogeneous but unified UNIX-like environment** spanning Linux servers, macOS on ARM-based Apple Silicon, and cloud platforms. Windows increasingly functions primarily as a client interface rather than a core scientific backend. **Python** now dominates research-level processing, data discovery, machine learning, and visualization. **C, C++, and FORTRAN** persist as performance-critical kernels, real-time system components, and legacy modeling codes, typically wrapped by higher-level interfaces. **Java** remains important in large enterprise-style processing systems and some real-time frameworks, while **MATLAB** continues in education, signal-processing research, and legacy laboratory workflows.

By the early **2020s**, seismology had fully entered the era of **continuous, automated, global-scale analysis**. Data volume is no longer the primary limiter of scientific ambition. Instead, the dominant challenges have become **computational cost,**

software complexity, algorithmic transparency, model bias, and the long-term sustainability of cyberinfrastructure. The field now operates as a truly **planetary-scale, real-time, computational Earth observatory**, built upon five decades of cumulative advances in computing hardware, networking, software, and digital data systems.

Looking Ahead: From Machine Limits to Human Limits

Across six decades of computational seismology, the dominant constraint has shifted repeatedly—but always in the same direction. In the **1960s and 1970s**, the limiting factors were **hardware scarcity, memory, and CPU time**. In the **1980s and 1990s**, constraints moved toward **storage, telemetry, and network bandwidth**, shaping how much data could even be retained. By the **2000s and 2010s**, those constraints largely dissolved, replaced by challenges of **software integration, data management, and reproducibility**. In the **2020s**, computing power, storage, and connectivity have become abundant; the primary bottlenecks are no longer machines, but **people**.

Modern seismology now operates in an environment where **planetary-scale data and computation are routine**. Continuous global waveform archives, real-time processing pipelines, and machine-learning models operate largely automatically, often beyond the direct intuition of any single analyst. The challenge is no longer how to compute something, but **what to compute, how to trust it, and how to sustain it over time**.

As a result, the most critical skills for the next generation of seismologists are shifting. Technical competence remains essential, but it is increasingly complemented by the ability to **design robust workflows, reason about algorithmic behavior, validate automated systems, and communicate uncertainty clearly**. Understanding how software is built, tested, documented, and maintained is now as important as understanding the underlying physics. Reproducibility, transparency, and interpretability are becoming scientific requirements rather than optional best practices.

At the same time, increasing levels of automation—particularly through machine learning—raise new questions about **bias, generalization, and failure modes**, especially in high-stakes operational contexts such as hazard monitoring and early warning. Future progress will depend not only on faster algorithms or larger models, but on **careful integration of human expertise with automated systems**, ensuring that machines amplify scientific judgment rather than replace it.

Looking forward, the trajectory is clear: continued movement toward **higher levels of abstraction**, where scientists spend less time managing files, machines, and infrastructure, and more time framing questions, interpreting results, and making decisions. The history of computational seismology is ultimately the history of **deciding**

what humans should no longer have to think about—and what they must continue to think about carefully.

In this sense, the future of seismology is not defined by any single technology—cloud computing, artificial intelligence, or new sensors—but by how effectively the community can **build sustainable, transparent, and trustworthy computational systems** that serve both scientific discovery and public safety.