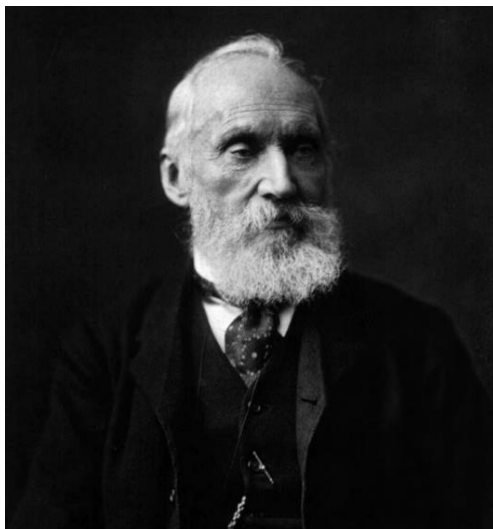


From Telegraph Cables to Artificial Intelligence: Unlocking the Power of Data

By Thomas O'Haver, Professor Emeritus, Department of Chemistry and Biochemistry



In 1882, when our University was known as the Maryland Agricultural College, Johns Hopkins University in Baltimore invited William Thomson - who would later become Lord Kelvin - to present a series of lectures on physics. You may recognize his title as the name of the absolute temperature scale used to this day by scientists.

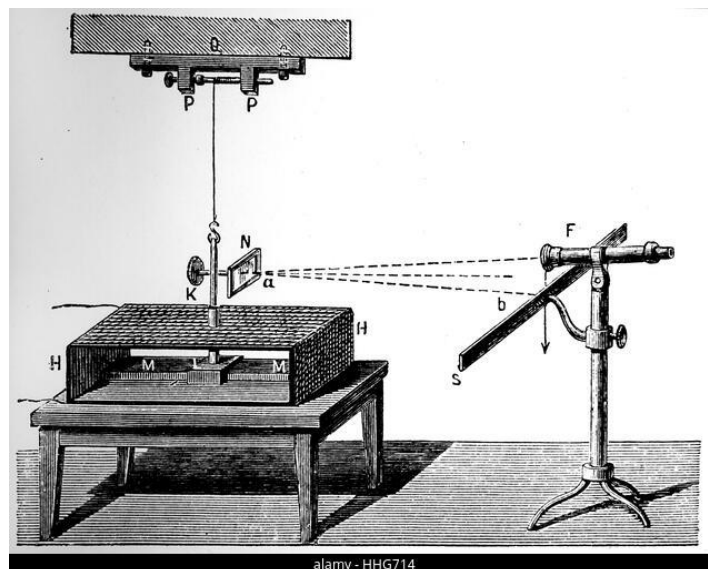


That title was derived from the River Kelvin that ran close to his lab in Glasgow, Scotland, where he was a professor for many years (and where I, coincidentally, spend a year on sabbatical in the 1980s, almost exactly a century after his tenure there).



Thomson's lifetime had bridged the transition between classical and modern physics; that lecture in Baltimore in 1882 occurred before the discovery of radioactivity, or of the electron, or of atomic structure. But although he remained largely attached to the comfort of classical mechanics and thermodynamics, he made important contributions to the understanding of heat, energy, and electrical phenomena. Moreover, he was a practical man, who created useful innovations in marine navigation, compasses, and telegraphy.

Thomson was involved in the struggle to accomplish undersea telegraphy in the 1850s, when the first cables were being laid across the channel between Dover and Calais, and then across the Atlantic between Britain and Canada. Those long cable runs resulted in weak and unreliable signals at the receiving end, creating what was arguably the *world's first electrical communication challenge*. The telegraphers wanted to solve the problem by increasing the signaling voltage, which they did until the voltage was so high it shocked a worker and ultimately burned out the cable, rendering the whole thing useless. There was no way to fix it, so a new cable had to be laid across the ocean. As a solution for that problem, Thomson invented a highly sensitive telegraph receiver, called a "mirror galvanometer", which allowed the signaling voltage to be kept safely low.



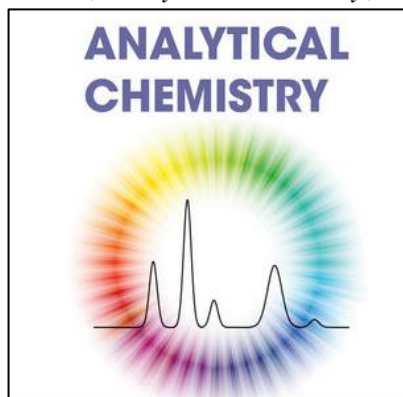
alamy - HHG714

Mirror Galvanometer

Throughout his career, Lord Kelvin was obsessed with the importance of quantitative measurement, at a time when there was not even basic agreement on the *units* of the measurement of electricity and temperature. It was he who wrote this famous injunction:

“When you can measure what you are speaking about, and express it in numbers, you know something about it....”

That focus on measurement, especially sensitive measurement, is shared by my own area of chemical science, *analytical chemistry*, which concerns itself with the sensitive measurement of chemical substances.



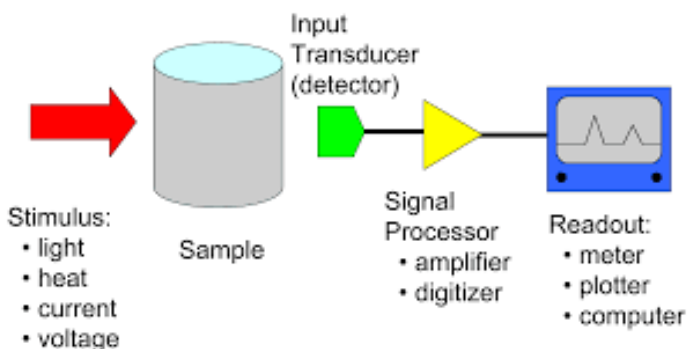
Fast forward 100 years....

Lord Kelvin could not possibly have imagined how far science and scientific measurement would come in 140 years. Unlike the stereotype of science laboratories in movies, which are invariably cluttered with flasks containing colorful bubbling liquids, by the 1980's modern science laboratories in all branches of science were cluttered with sensitive electronic measurement instruments that often contained, or were connected to, computers.



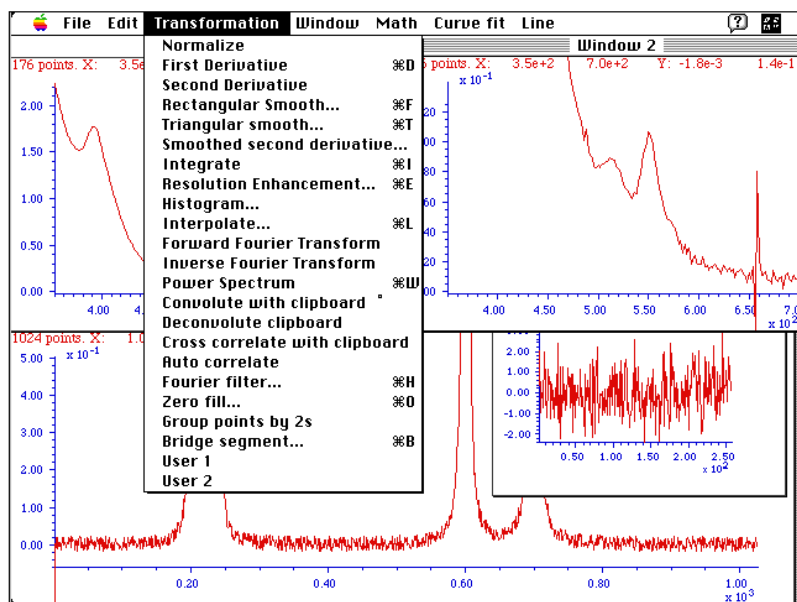
When microprocessors and personal computers made their first appearance in the 1970s, I became fascinated by their potential applications in the laboratory to acquire, process and extract measurements from *numerical scientific data* (*i.e.* “signals”), using what would otherwise have been laborious

mathematical techniques requiring the use of cumbersome remote “mainframe” computers. These techniques can transform signals into more useful forms, detect and measure peaks, reduce noise, improve the resolution of overlapping peaks, compensate for instrumental artifacts, test hypotheses, optimize measurement strategies, diagnose measurement difficulties, and visualize complex signals and decompose them into their component parts. This can often make



difficult measurements easier by extracting more information from the available data, especially when used on small computers right in the laboratory where they can provide immediate feedback.

I did a little work in this area that resulted in some papers in the 1980s and 90s and even a couple of commercial products. In the decades since, the hardware and software have become enormously more capable, smaller, less expensive, and more widely applied far beyond the physical sciences, arguably outpacing the educational support for their applications.



My first attempt at writing this kind of signal processing software was an all-in-one program, coded in *Pascal*, that I called **SPECTRUM**, an acronym for **S**ignal **P**rocessing for **E**xperimental **C**hemistry **T**eaching and **R**esearch/ **U**niversity of **M**aryland, written in 1989 for the then-new mouse-controlled graphical user interface. This worked well enough for teaching purposes, and even won a couple of awards at the time, but for *research* purposes the program proved inflexible and insufficiently powerful.

After I retired from teaching in 1999, I thought I could expand that work into a

volunteer educational outreach project, using the Internet to reach beyond the walls of the University. My objective is to help students and researchers with computer-based processing of numerical scientific data, by creating a “toolchest” of modular software functions, such as *smoothing*, *filtering*, *peak detection*, *deconvolution*, *curve fitting*, *regression*, *denoising*, etc., along with a supporting [website](#) and a comprehensive handbook (now grown to 533 pages), with email-based support and consulting. Based on web “hits” and emails, the interest in such applications has been much wider than I had anticipated, both in terms of geographical diversity and in the wide range of research areas, far beyond my own experience. A central goal therefore became making my work more accessible to all by simplifying the treatment and by lowering costs.

The entry point is the following URL: <https://terpconnect.umd.edu/~toh/spectrum/>

Reducing friction: Making it accessible and affordable

All of this is relatively easy for those of us in rich countries, at modern high-tech educational or company campuses, in labs with the latest expensive software installed and with experienced workers or students ready to help us. It’s harder for others in less well-endowed locations, with smaller budgets, whose first language might not be English, or whose math preparation might not be as complete. The last time I counted, 162 countries had accessed by website, including some that are relatively isolated by geography, by government policy, or by war (Faroe Islands, Cuba, North Korea, Myanmar/Burma, Afghanistan, Syria, Iraq, Ukraine). Doing science in some countries is a serious challenge.

So, to make my work more universally accessible and affordable, I have taken several measures:

Writing. The text of the project is written in **American English**, in a plain style, devoid of unnecessary jargon, idioms, figures of speech, metaphors, cultural references, sarcasm, irony, and humor, all of which might be difficult for those with limited English skills or for machine translators. The reading level is **11th grade**, according to several automated readability indexes.

The standard textbook treatments of these subjects confront the reader with page after page of abstract math, which some might find frightening. My treatment is **pragmatic and much less formally**

mathematical, to make it less intimidating for those without a solid math background. In some parts, I do use some basic algebra, or even a bit of elementary calculus and matrix algebra, but I rely more on *logic, practical examples, graphics, analogies, and animations* rather than abstract math. Many concepts are explained repeatedly in different settings and in different practical applications.

“Fresh wording and concrete images force us to keep updating the virtual reality display in our minds.”
Steven Pinker, from *The Sense of Style*.

My project is free, including email-based consulting. I reject offers of paid consulting. Because I am retired, I am no longer building my curriculum vitae; I don't need more papers in “high-impact” journals; I don't want money or prizes for this; I propose to do it altruistically and see what happens.

It is suitable even for poorly funded laboratories. My software is “Open Source” and can run on commercial as well as freely downloadable software platforms and on modest desktop or laptop computers, as well as on portable hardware (e.g., Internet connected tablets, smartphones, and even some single-board computers like the *Raspberry Pi*). The entire project, including the book and all the scripts and functions, can be stored on a small flash drive or SD card, for transfer to people or locations with slow or unreliable Internet access.

Multiple in available formats: The text portion of my project is available in several digital formats and on paper (through Amazon's CreateSpace self-publishing program). All have a detailed table of contents. The digital versions include a complete citation list. (The Microsoft Word 365 DOCX retains the familiar book-like format while showing the animated graphics on the page).

Teaching and writing on the Internet.

Google (or any search engine) looks at the entire internet, irrespective of the academic specialization, in contrast to the formal literature which is **cordoned into specialties**. *Google* is not constrained by traditional institutional boundaries.

For example, why would a *neuroscientist* know anything about my work, or a *cancer researcher*, or an *economist*, or a *linguist*, or a *music scholar* for that matter? I published in analytical chemistry journals that they certainly don't read. But all those types of researchers have not only found my work and downloaded it, but have *cited in their research publications*.

The fact that I am writing on the Internet, and that a search engine might be the main mechanism by which my work would be found, must influence the way that I write. Whereas in my classes at the University I had some idea of the background of the students, and I could expect a certain minimum background in mathematics and science generally and, in my case, perhaps even in chemistry. But the world of people who might find my work on the Internet via a Google search is far wider than that. Students attend classes because they *need that course for requirements*, whereas people visit web sites because they *are searching for that information/software*.

Because of this, my book is instructional, not scholarly, and it is titled “pragmatic” (which means “Relating to matters of fact or practical affairs, often to the *exclusion of intellectual or artistic matters*; practical as opposed to idealistic”).

I've had lots of feedback from users, via email, social media, search engine terms, with thanks, questions, corrections, requests, suggestions for additions, etc. I re-read and re-write sections of my book, often prompted by questions from readers. I edit for (hopefully) improved clarity. Most usefully, many users have sent *attached examples of numerical data* from their experiments, allowing me to apply my tools to types of data that I could not have possibly experienced in my research career.

Contact with clients.

My communication with clients is exclusively via email. I know only what they tell me voluntarily, so I typically do not know their position or educational level. Some do provide interesting details about their research; others just send me numerical data with no information about what it means.

Even though video conferencing platforms are now widely used around the world, I politely deflect requests for phone or video conversation, for several reasons:

1. The store-and-forward nature of email is better suited to communication across time zones.
2. There is evidence that people tend to make incorrect judgments about other people based on their physical appearance and behavior (Malcolm Gladwell, “*Talking to Strangers*”)
3. Writing is less spontaneous than speaking and encourages more thoughtfulness.
4. Those with limited English skills can take their time composing a message and can even use a machine translator.
5. I have modest age-related hearing loss.

Is there really a need for all this?

Isn't software already included in every modern scientific instrument hardware purchase? Yes, at least for those who are using conventional instruments in standard ways. But many scientists are working in new research areas for which there are no commercial instruments, or they are using modifications of existing systems, or they are building completely new types of measurement systems for which there is no software. In some cases, the commercially provided software is inflexible, inadequately documented, or hard to use.

But writing one's own software from scratch is not always the solution. Not every researcher or science worker likes programming, or has time for it, or is good at it. Hired programmers typically may not understand the science and in any case sooner or later move on and no longer maintain their code, leaving you with a mess to clean up.

Software selection criteria

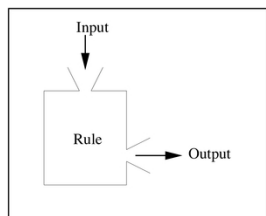
Writing applied computer code is a *craft*, not a science. It's a bit like the modern equivalent to the glassblowing and model shops that science departments used to keep, back when I was a student. But because I am retired, I have the freedom to indulge in coding, just like any other craft.

Rather than creating self-contained stand-alone programs, my goal is to create a collection of *modular blocks of computer code*, called “functions”, a bit like virtual *Lego* bricks that can be snapped together to build different structures. The object is to make it easier for users to construct and optimize their own specialized scripts by copying and pasting sections of code.

This is how you assemble a customized entertainment or home security system using separate commercial components. You start with a collection of separate components, usually from different manufacturers, such as smartphones, tablets, computers, TVs, security cameras, doorbells, thermostats, printers, earphones, speakers, etc. Then you connect them using standard cables and/or Bluetooth and Wi-Fi connections, *all without knowing about the internal design of each component*, using only the input and output sockets. Even then, the task is not trivial; an overall system plan is vital.

Software functions receive one or more numerical “inputs”, performs a calculation, and provides one or more “outputs”. A simple example is the square root button on a calculator. You type in a number (the *input*), press the square root button (the *function*), and the result (*output*) is displayed.

It's similar in programming a spreadsheet, you type “=sqrt()” into a cell and insert into the parentheses a number or a reference to another cell or range of cells.



This works the same in function-based computer languages such as *Python* or *Matlab*, but the functions there can be designed to perform far more complex tasks, which can be used later *without contending with the internal complexity* of the functions, only dealing with the input(s) and output(s). In computer languages, *you can write your own functions*, which can incorporate earlier-defined functions and can be combined to perform arbitrarily larger tasks, or you can borrow functions created by others. This type of structure not only makes it easier to program, but it also makes it easier to break down an existing program into its simpler and more easily understood component parts and to test the parts independently.



The concept of modularity extends even to chemistry. The 2022 Nobel Prize in Chemistry was won for the development of a functional form of chemistry – *click chemistry* – in which molecular building blocks snap together quickly and efficiently.

I have created a toolchest of over 200 software modules (demonstration scripts, command-line functions and Live Scripts) that are relevant to scientific data processing and that can be *called by the user as needed to construct unique customized programs for their own purposes*.

Software platforms: To make this as widely accessible as possible, I use existing popular development environments with multiple contributors: mostly Matlab, Octave, Python, Excel and Open Office Calc spreadsheets. These run on PC, Mac, Unix, even on some portable/miniature/expendable devices like the *Raspberry Pi*. I try to balance *cost, speed, ease of use*, and a *gentler learning curve*. Matlab is the fastest of these computationally, but commercial versions are expensive. Fortunately, Matlab and Excel are widely available via site licenses at many university, industrial, and government research campuses. Moreover, open-source “clone” versions that are *mostly compatible* and are *freely downloadable*. Many user-contributed functions and code examples are available online for both Matlab and Python.

I add help and examples that are *built into the software*, accessed by typing “help <function name>”, after which the examples can be copied and pasted into the command line to run them immediately. I also provide many “demos”, which are full operational demonstration scripts that load an example of real or synthetic example data and then call one or more of my functions to demonstrate a realistic application. My materials have lots of examples of applications of my functions, including videos and *animated GIFs* that will play on any web browser and in “.docx” format when viewed in *Microsoft Word 365* (but not most PDF viewers), such as the examples below.

Both Matlab and Python have more modern *interactive alternatives* to conventional textual scripts. [Live Scripts](#) in Matlab are interactive documents that combine code, interactive controllers, output, and formatted text in a single environment called the Live Editor, which allows you to start with a conventional script and simply insert interactive elements into it. Python has [Jupyter Notebooks](#) which are used to create an interactive narrative around your code. Both make it easy to create interactive documents with graphical user interface devices such as file browsers, pull-down menus, check boxes, and sliders to adjust numerical values interactively. The development path is far easier than for conventional stand-alone “apps”. I have developed Live Scripts tools for [smoothing](#), [deconvolution](#), [peak detection](#), and iterative [peak fitting](#), a portion of which is shown below.

Peak Fitting Tool

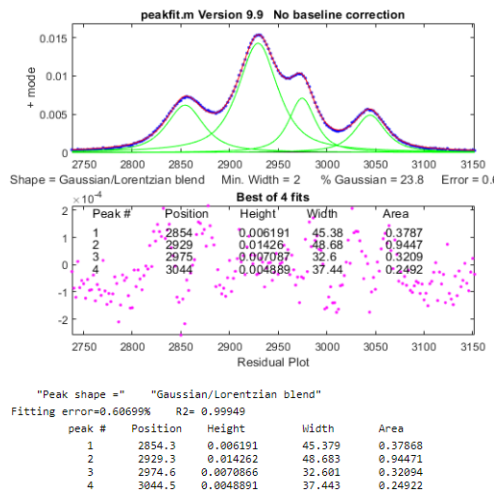
with a selection of different peak shapes and optional symmetrization of exponentially-broadened peaks. (Requires my `peakfit.m` function in the Matlab path). Tom O'Haver 5/9/2023

clear

`Open data file` file= uigetfile('*.csv;*.xlsx'); % Click this button to load data
mydata=xlsread(file); disp(file); % from disk in xlsx or csv format.

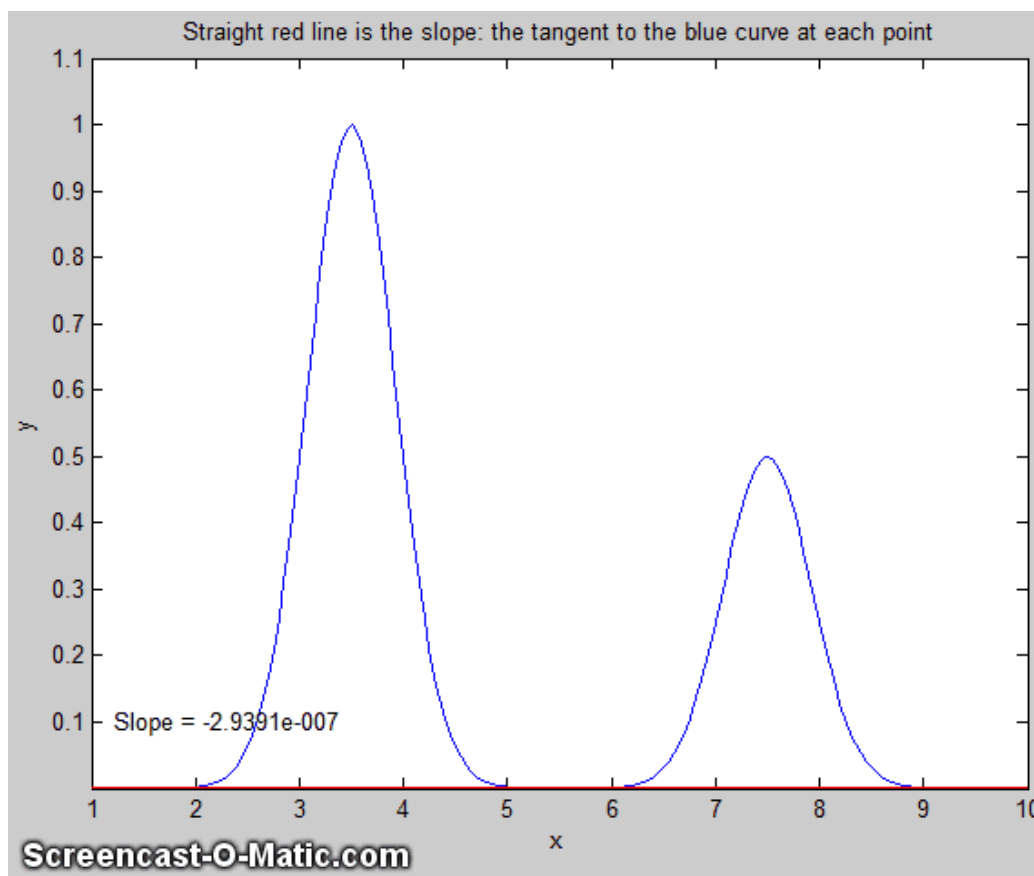
```
startpc= 37 ; % Percentage of data points to start data selection  
endpc= 57.6 ; % Percentage of data points to end data selection  
  
PreProcess=  ; % Optional data pre-processing before curve fitting  
SymmetrizeFactor= 0 ; % "de-tailing" for skewed peaks exponentially modify  
SmoothWidth= 1 ; % Smooth only if using symmetrization  
NumPasses= 3 ; % Number of sliding average passes (1-5)  
VerticalShift= 0 ; % Vertical signal shift, percent of maximum signal.  
  
% Controls for least-squares peak fitting  
FitPeaks=  ; % Perform curve fit  
PeakShape= (Gaussian/Lorentzia...)  
NumPeaks= 4 ; % Number of peaks in fitting model (1 to 10)  
NumTrials= 4 ; % Takes best of "NumTrials" curve fit trials (1 to 10)  
extra= 23.8 ; % Shape parameter for variable-shape peaks (EMG, G/L blend, et
```

HepteneTestData.csv



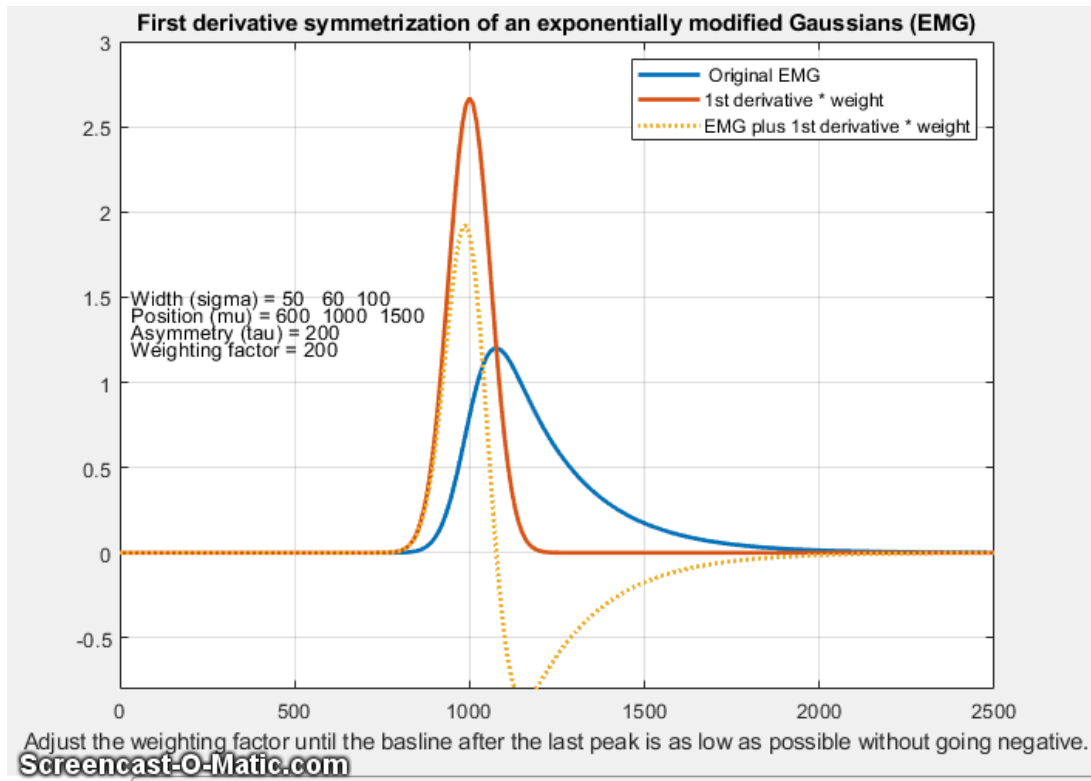
Instructional animated graphics examples

An elementary pre-calculus concept: The *first derivative* at each point on the curve is the slope of the line that is tangent to the curve at that point.



[Click for animation](#)

One practical application of the 1st derivative: A peak that is broadened and skewed by an *exponential process* (blue line) can be restored to its original shape (red line) simply by adding a portion its *first derivative* (dotted line) to the distorted signal. [Double-click for animation.](#)



The mathematical proof is arguably harder to understand:

$$\exp(-x^2/2) = -x \cdot \exp(-x^2/2);$$

$$z' = -1/\sqrt{2}$$

$$\begin{aligned} \frac{EMG'(x)}{k} &= (G(x)E(x))' = G'(x)E(x) + G(x)E'(x) = -xG(x)E(x) + G(x)\left(2zE(z) - \frac{2}{\sqrt{\pi}}\right)z' \\ &= -xG(x)E(x) + G(x)\left(2\left(\frac{-x + \frac{1}{\tau}}{\sqrt{2}}\right)E(x) - \frac{2}{\sqrt{\pi}}\right)\frac{-1}{\sqrt{2}} \\ &= -xG(x)E(x) + xG(x)E(x) - \frac{G(x)E(x)}{\tau} + \frac{G(x)}{\sqrt{\pi/2}} = -\frac{G(x)E(x)}{\tau} + \frac{G(x)}{\sqrt{\pi/2}} \end{aligned}$$

We multiply both parts by $k \cdot \tau$

$$\tau \cdot EMG'(x) = -k G(x)E(x) + k\tau \sqrt{\frac{2}{\pi}}G(x) = -EMG(x) + hG(x)$$

And thus





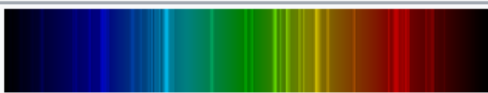


$$\tau \cdot EMG'(x) + EMG(x) = GSS(x)$$

Courtesy of Yuri Kalambet, Ampersand Ltd., Moscow, Russia, "Reconstruction of exponentially modified functions", 2019. DOI: 10.13140/RG.2.2.12482.84160.

Practical examples, with experimental data sent to me by users

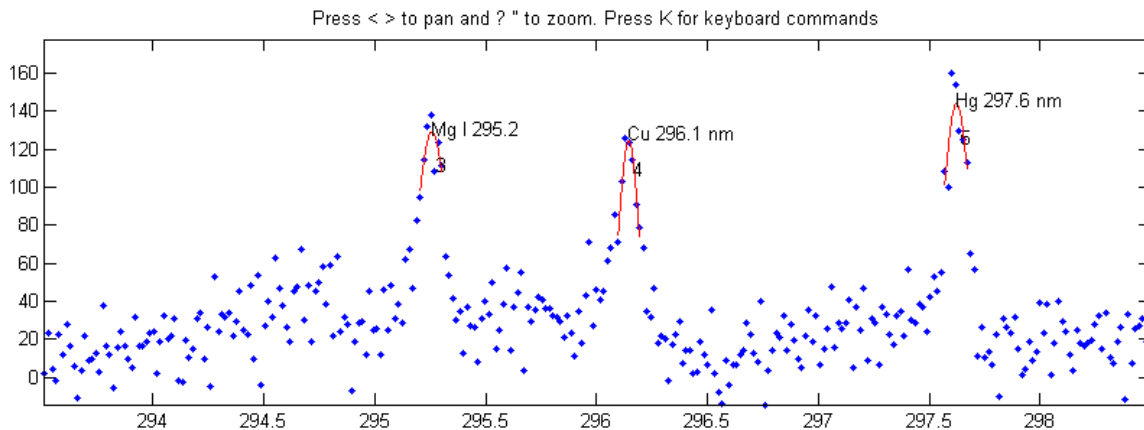
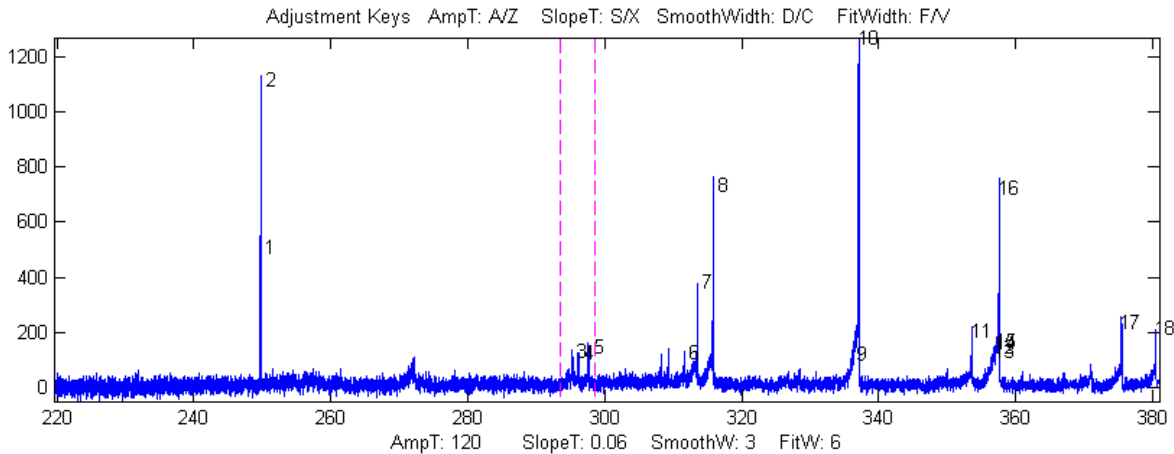
Example 1: Automated spectroscopic trace element analysis

Background: When vaporized, every chemical element has many narrow spectroscopic peaks, called *lines*, at *very specific x-axis positions*. The systematic attribution of spectral lines to chemical elements began in the 1860s (as well as the discovery that the stars were composed of the same elements as found on earth). Here's are examples for the first few elements.

V·T·E				Spectral lines of the chemical elements
Element	Z	Symbol	Spectral lines	
hydrogen	1	H		
helium	2	He		
lithium	3	Li		
beryllium	4	Be		
boron	5	B		
carbon	6	C		
nitrogen	7	N		
oxygen	8	O		
fluorine	9	F		
neon	10	Ne		
sodium	11	Na		
magnesium	12	Mg		

This spectrum is from a sample of black soot. To *automate* the detection and identification of elements via their spectra, the software must:

- (1) scan and digitize the spectrum,
- (2) detect the peaks that meet specified criteria,
- (3) determine their x-axis position accurately, and
- (4) look up which element is close enough to that x-value.



Name	x Position	x Error	y Amplitude
'Mg I 295.2'	[295.2]	[0.058545]	[129.27]
'Cu 296.1'	[296.1]	[0.045368]	[124.6]
'Hg 297.6'	[297.6]	[0.023142]	[143.95]

Function from my toolchest: (idpeaks.m, 225 lines of code, which is part of the interactive function ipeak.m, 6079 lines)

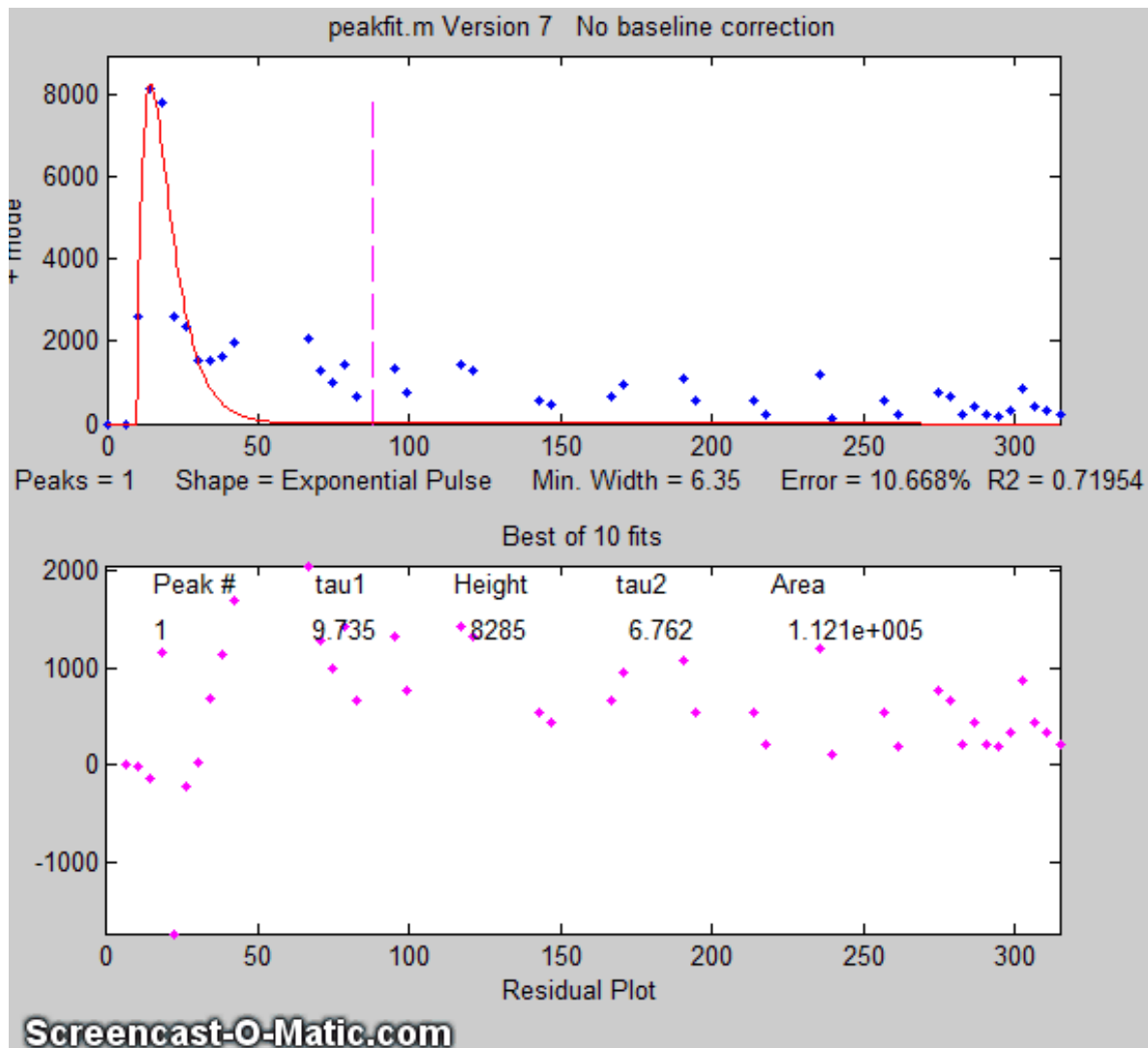
This technique can be applied not just to spectroscopy, but to *any data set with peaks whose x-axis positions are reliably correlated with identity* (e.g. chromatography, drug detection, etc.) The user simply needs to provide, as an input to the function, a specific table of known peak positions and their identities.

Example 2: Following the evolution of a variable peak signal in real time

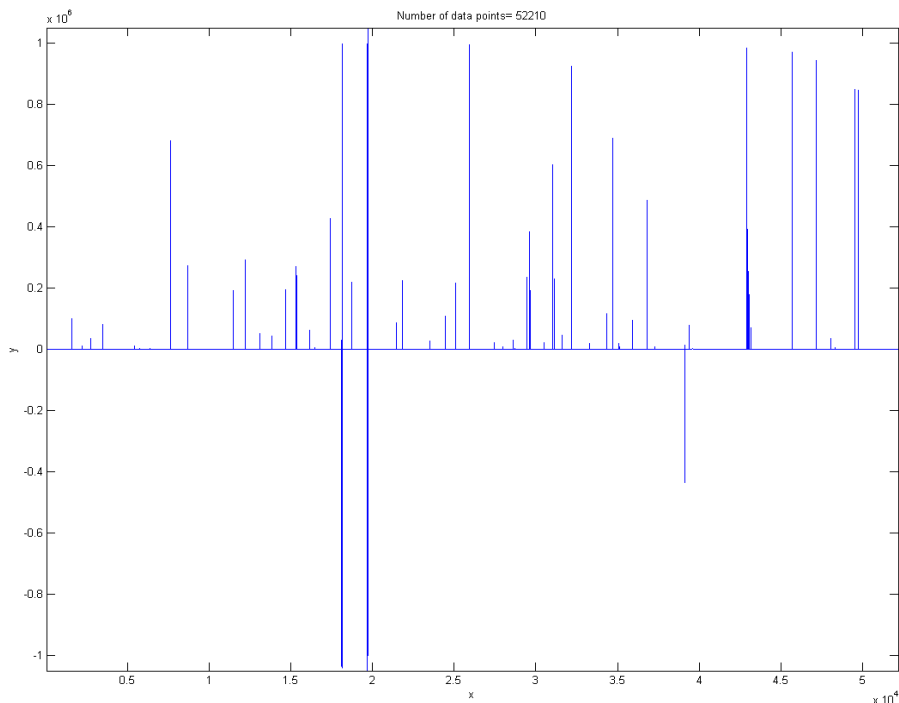
In this example, a client wanted to know how a series of pulses from his experiment conformed to an exponential pulse of the form:

$$y = \exp(-\tau_1 \cdot x) \cdot (1 - \exp(-\tau_2 \cdot x))$$

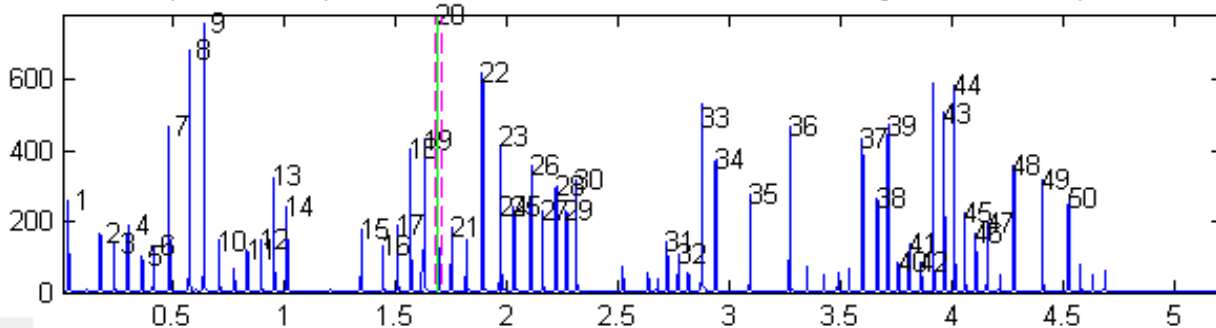
I did this by placing my iterative curve fitting “peakfit.m” function in a loop. On contemporary small computers, curve fitting is sufficiently fast (~0.05 seconds) that can often be used in “real time” (Function from my toolchest: peakfit.m, 4026 lines)



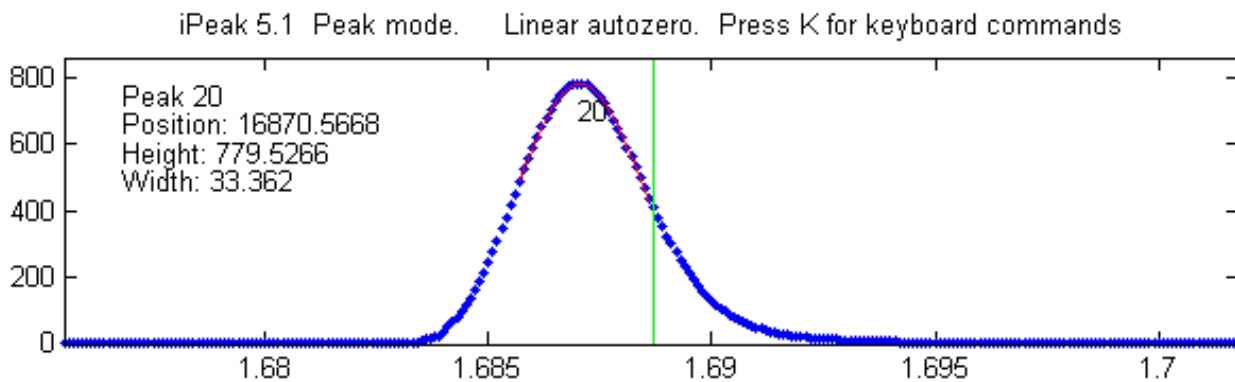
Example 3: “Drilling down” to find the real signal peaks that are invisible and completely buried in a forest of much larger (x1000) artifacts.



Step 1: Eliminate the artifacts (sliding “spike killer” function, 6 - 8 lines of code, which “patches over” spikes by linear interpolation before and after.)



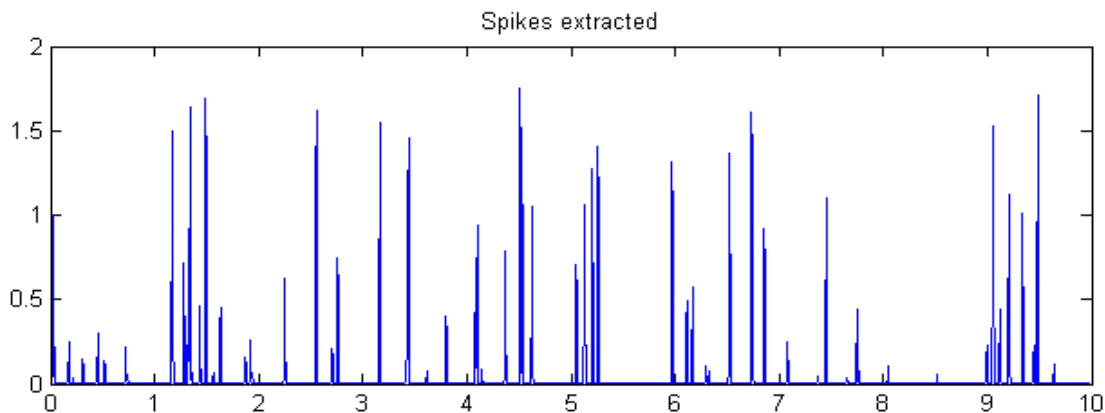
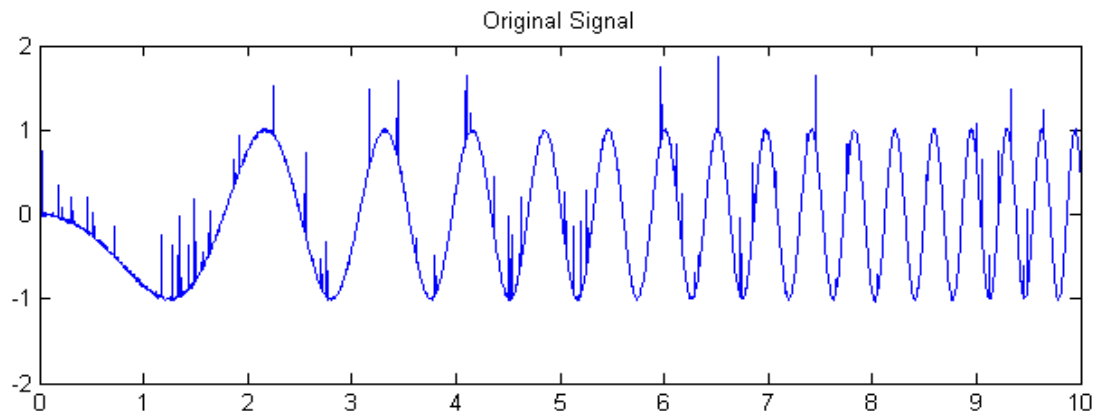
Step 2: Measure the “good” peaks by local curve fitting (“ipeak” function, 6070 lines)



Example 4: When the sharp spikes *are* the signal.

A study of beach erosion caused by wind-blown sand.

The researcher mounted tiny microphone elements on poles along the beach, recorded the sound, and attempted to distinguish the sharp ticks caused by wind-blown sand grains from the shorebird calls and whistling wind. *Fourier filtering* did the trick (**Function from my toolchest: ifilter.m, 442 lines**)

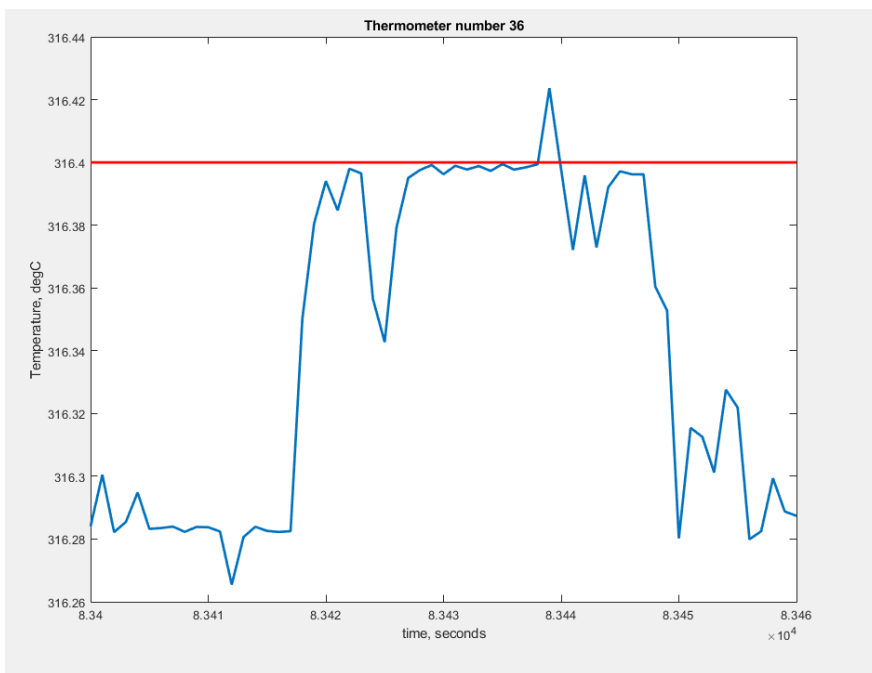


The originator of the Fourier Series and the Fourier Theorem, Joseph Fourier was the son of a tailor, orphaned at age 12, and educated by Benedictine monks. He lived through and survived the French Revolution and eventually accompanied Napoleon Bonaparte on his expedition to Egypt in 1798, where some of Napoleon's soldiers uncovered a buried stone slab with an inscription written in three ancient languages, in a little town whose name they mispronounced as "Rosetta".

Example 5: January, 2022, Czech nuclear power station. www.ujv.cz

The reactor had 210 separate thermal sensors, each needing to be monitored once each second continuously, 24/7. They needed a program that would sound a warning if any of the sensors exceeded a set temperature rise limit. (50 lines of code). An engineer sent me one day's worth of typical data (210 x 24 x 60 x 60 data points).

Hours	minutes	seconds	thermometer	Temperature, degC
9	512	30720	4	316.46
9	515	30900	4	316.57
9	526	31560	36	316.51
9	528	31680	4	316.49
9	556	33360	36	316.51
10	604	36240	36	316.41
12	705	42300	36	316.46
12	735	44100	4	316.44
.				



In February 2022, Russia invaded Ukraine. In August, when the Russians reached the Zaporizhzhia power plant, the Czech engineer wrote to describe their problems and safety precautions they were taking, and he added this note:

“Regarding refugees from Ukraine, the situation is better than we all expected in the Czech Republic.... The most surprising fact is that all refugees are able and willing to find a job within a few weeks! So they can rent an apartment and start living "normally". Ukrainians are very brave and willing to work very hard.”

Outcomes

A. Did my efforts make my book more understandable and accessible?

Here are a few verbatim comments from hundreds of unsolicited emails:

"... It's **by far the best** document I've found about signal processing online."

" **It is the best source in the entire internet** ... on peak related techniques and analysis."

"...the **most understandable** documentation I've ever encountered!"

"It is as good as, if not **better than, any similar books available** in the market".

"...a **clear, organized, accessible, and intuitive** resource...."

"I ... have never seen before such a **complete approach** towards tackling almost all the analysis problems from scratch...."

"...the **most extensive set of tools I've ever found** for curve fitting and signal processing!"

"I'm **amazed by the thoroughness** of not just the explanations, but also the codes and documentation within."

"**It's simply phenomenal!** "

"I appreciate the fact that your approach is 'applied' and **does not resort to complex mathematics where not needed**"

"...your work was a **godsend**."

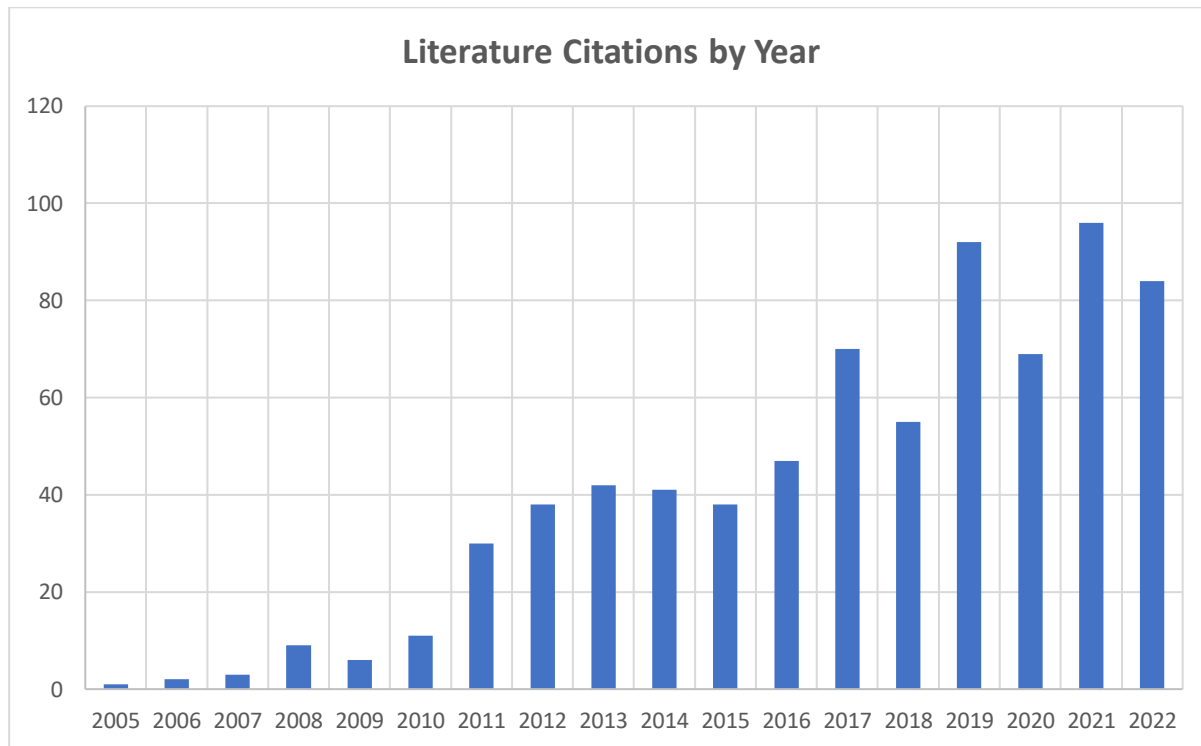
"... *surely **the way into heaven** is to post excellent software online for others to use to do science.*"

"...it **inspires me to make my own contribution** to science and education."

"I'm really enjoying and also being **greatly inspired** by your ... project"

"Your program ... is like falling out of a tree and landing in a soft couch complete with a book and a good reading light."

B. As of Dec 2022, I had received a total of 696 citations in research papers



Some interesting applications

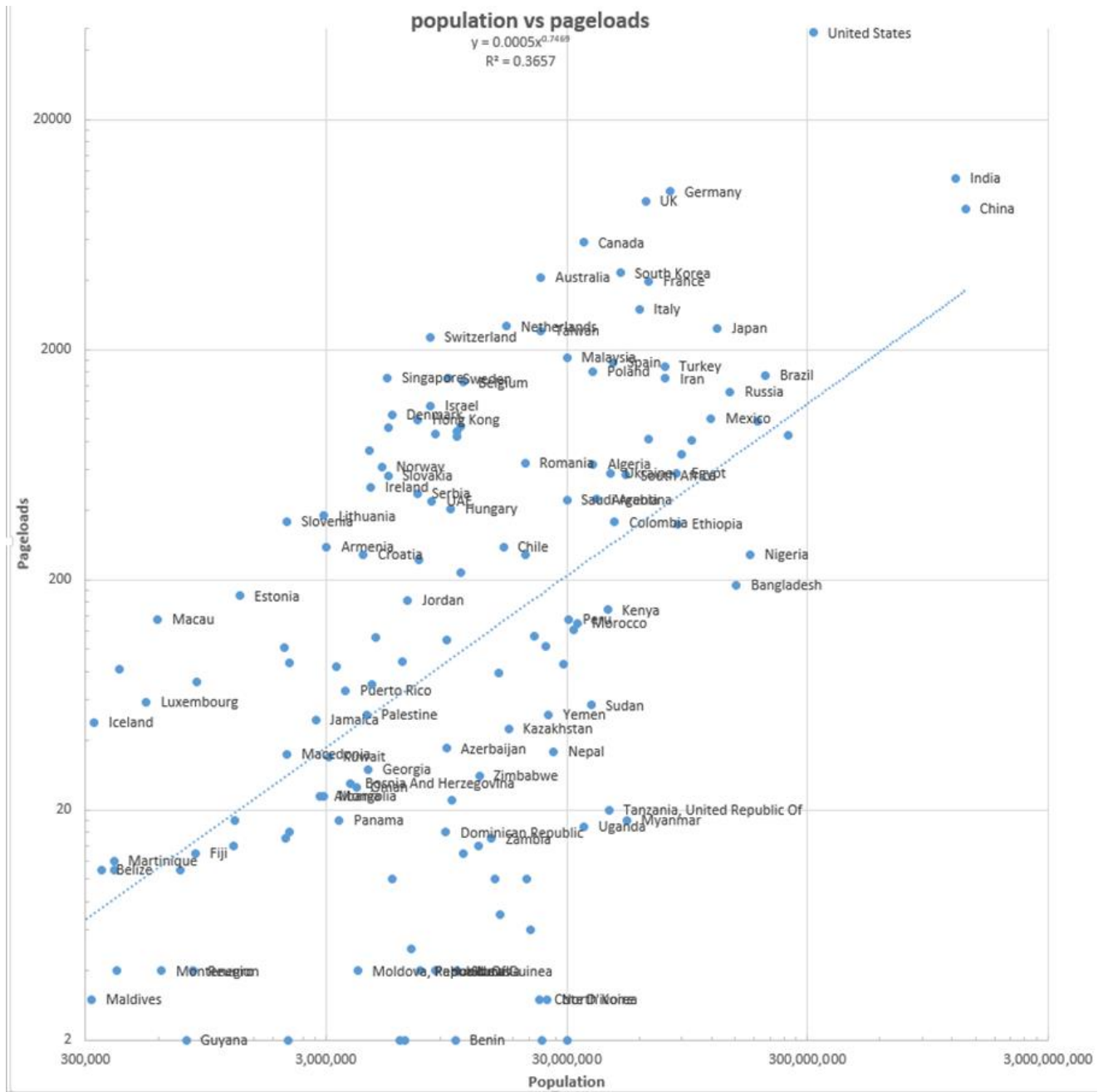
Selected titles of citing publications, 2007 - 2021:

TNT-based explosive detection
Chemical release wind measurements
Red wines subjected to accelerated ageing
Concrete strength monitoring
Music information retrieval
Deep-sea bamboo corals
Assistive technology in an ALS patient
Heart sound classification
On-comet attitude determination
Tremor signals in Parkinson's disease
Blood flow imaging through the skull
Find the numbers of parking spots
Forest reflectance characterization
Classification of speech sounds in dialects
High-speed rail suspension monitoring
Neural responses in progressive aphasia
Seismic response of gas pipelines

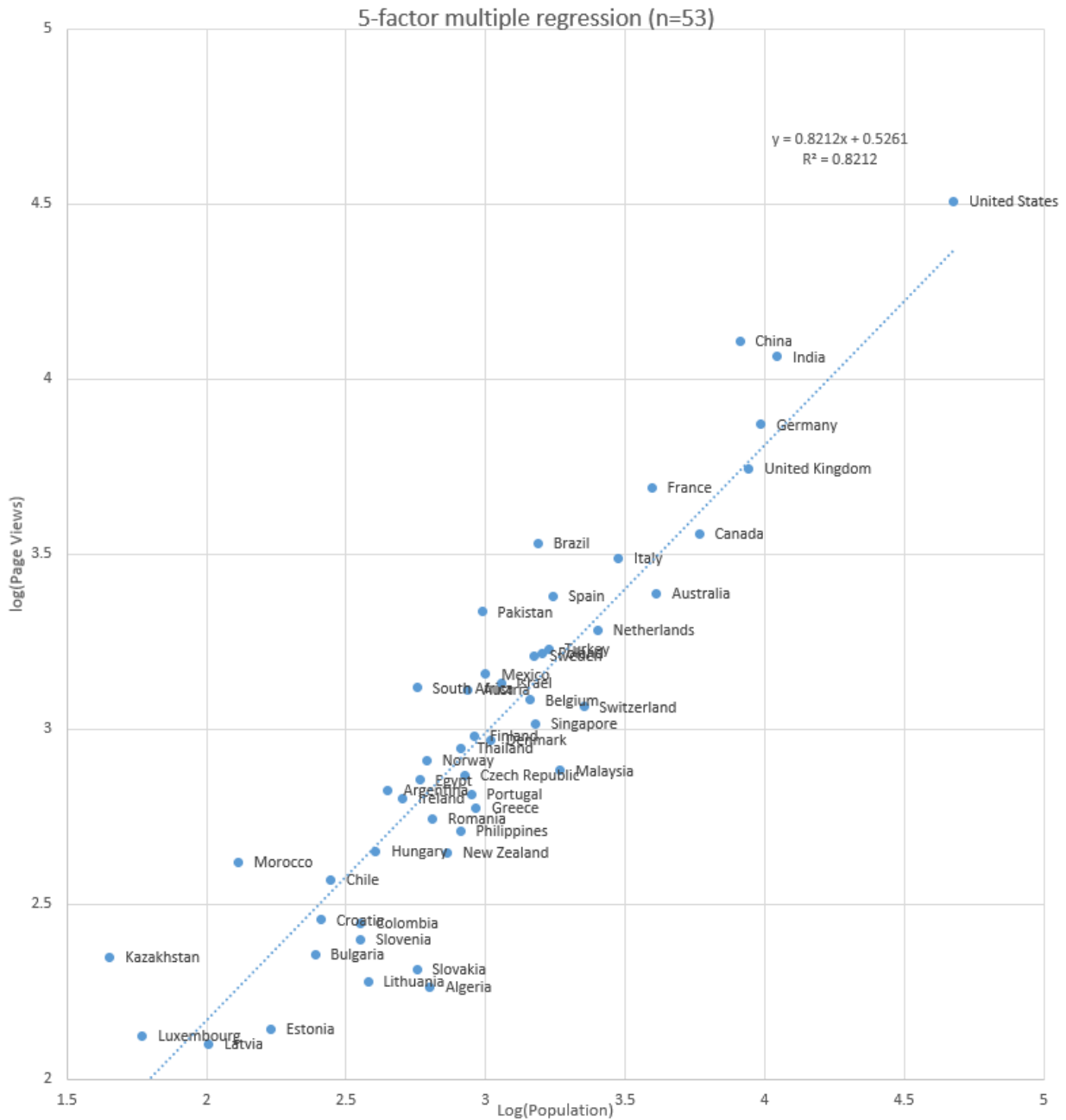
Identification of illicit opium cultivation
Characterization of lunar soils
Analysis of biofuel quality
Neural signals in autonomous prosthetics
Decomposition kinetics of tomato peels
Measuring patient activity level and posture
Microbeam radiation therapy
Cladding materials for nuclear reactors
Rainfall threshold for deep landslides
Body-worn vital sign monitor
Radioactivity in plant and soil samples
Quantifying tumor drug resistance
Eye blink detection
Olive oil characterization
Measurement of fentanyl in recreational drugs
Traffic-speed road surveys
Determination of anti-parkinson drugs

What explains the huge differences in the number of web hits between countries?

Here is a scatter plot of the number of hits on my web site by country as a function of the population of each country. It's clear that the *population* of the countries alone is not enough to predict the number of web hits. The R2 is only 0.36. Also, some countries (e.g., USA, Germany, UK, Taiwan, Switzerland, Sweden, Belgium) have scored *more* hits that expected for their population. Some countries score *far fewer*.



A better prediction of the number of web hits per country can be obtained by regression onto 5 factors: (1) the total population, (2) the number of English speakers, (3) the number of universities, (4) the number of Internet users, and (5) the amount of R&D spending. *N=53 countries for which all five factors are known.*



The Future

Where will all this go in the long run? As for my own retirement project, I will eventually move on to other projects, or pass on, and my work will fade away, to join the terabytes of forgotten material on the Internet that, unless specifically deleted, will presumably hang around forever, like moldy books on abandoned library shelves. But some of the techniques that I have written about may become part of the education of all science students, or they may be replaced by more sophisticated methods, or the processing of scientific data may be entrusted to artificial intelligences (AI). In fact, we are already surrounded by more AI than we probably appreciate. The following list is certainly incomplete:

Everyday applications of artificial intelligence

Grammar and spelling checkers, autocorrect, predictive typing;

Google predictive search algorithms;

Voice assistants like Siri, Cortana and Alexa;

Speaker-independent voice recognition;

Image recognition for cellphones face unlock and for tagging photos;

Photo editing; smart sharpening; deleting or changing backgrounds;

Generating drawings from a verbal description;

Travel map navigation; High-traffic warnings;

Self-driving cars; Automated vehicle safety;

AI adjusts traffic light timing based on vehicle speed and density;

Automatic recommendations on YouTube, Netflix, Pandora, Apple Music, etc.;

Stocking of regionally and seasonally ordered items on Amazon, food ordering sites, etc.;

Signature recognition in banking;

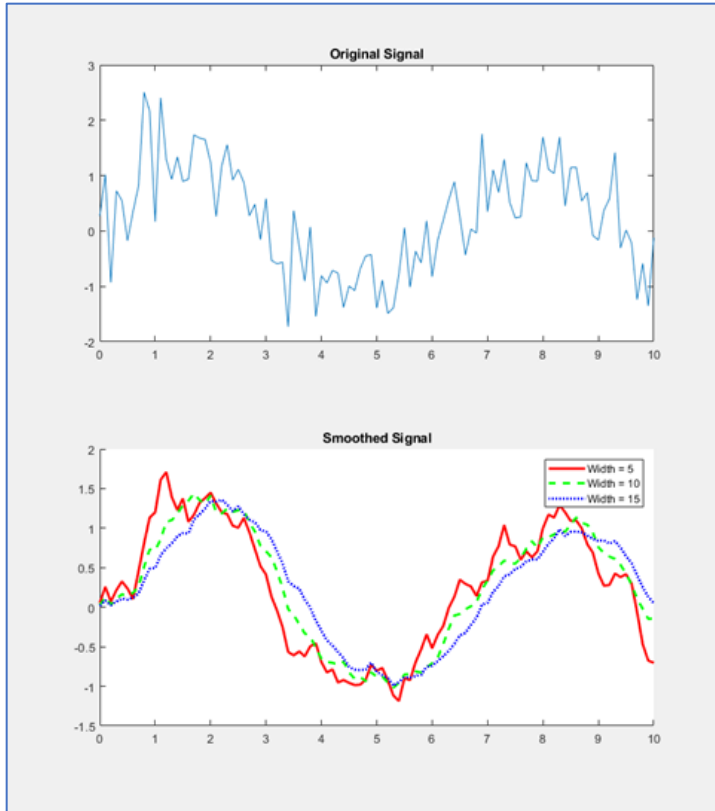
Predictions of repeat customers on Uber and Lyft;

Chatbots, such as ChatGPT.

Artificial intelligence “chatbots” like ChatGPT have considerable skills in language interpretation and writing; for example, it is quite good at suggesting possible titles for papers, talks, or proposals that you are working on. Just feed it all or a portion of what you have written. In fact, ChatGPT suggested the title of this document). It can also paraphrase and summarize, which can be useful for writing condensed summaries or abstracts. And it can create a proposed outline for a book or paper on any specified topic. But because it was trained by a group of computer scientists and engineers, it’s not surprising that it knows something about computers and their typical applications.

ChatGPT can even write simple computer programs. Here’s a simple example. Suppose you needed to create an illustrative graphic for a talk or class you are giving on data smoothing. Specifically, you need some code that “creates a plot with two horizontal subplots, the top one showing a signal with some peaks plus some random noise, and the bottom one showing the results of applying smooths of three different widths to that signal, each shown in a different line style and color, with a legend identifying each plot and its smooth width”.

This is not a very difficult problem; any programmer could bang this out easily in a dozen lines of code or so. But still, if you were in a hurry, or you were relatively new to programming, or you wanted this done in a language you were not very familiar with, you might look for an easier way out.



And, yes, as you have already guessed, an AI like ChatGPT can do that task easily, as shown by the ChatGPT-created graph on the left. It can write the code in any one of several computer languages, such as Matlab, Python, C/C++, R, Pascal, Fortran, HTML, JavaScript, Excel macros, *given only the above description and nothing more*. In addition, *it writes code in good style*, usually with explanatory comments for each line, appropriate indentation, and often includes examples of use and even warning under what circumstances the code may fail.

Of course, AI is not infallible; ChatGPT sometimes writes code that fails to do what you want or that does not work at all. It cannot read your mind; you must describe what you want clearly. (For more examples, see this page from my website: <https://terpconnect.umd.edu/~toh/spectrum/AI.html>)

The future is here, and it's getting closer.