MATH 1MP3

Introduction to Mathematical Scientific Computation

Winter 2026

B. Bourdin bourdin@mcmaster.ca

What is computer programming

From wikipedia:

- **Computer programming** or **coding** is the composition of sequences of instructions, called programs, that *computers* can follow to perform tasks. It involves designing and implementing *algorithms*, step-by-step specifications of procedures, by writing code in one or more *programming languages*.
- A **computer** is a machine that can be *programmed* to automatically carry out sequences of arithmetic or logical operations (computation).
- In mathematics and computer science, an **algorithm** is a finite sequence of mathematically rigorous instructions, typically used to solve a class of specific problems or to perform a computation.

Computer programming and mathematics

Ex. Quadratic formulas: solve $ax^2 + bx + c = 0$ $(a \neq 0)$:

set $\Delta = b^2 - 4ac$.
if $\Delta > 0$ then

$$x_1 = \frac{-b - \sqrt{\Delta}}{2a},\ x_2 = \frac{-b + \sqrt{\Delta}}{2a}$$

else if $\Delta = 0$ then

$$x_1 = x_2 = -\frac{b}{2a}$$

else
The problem does not admit a solution

Ex. Differentiation rules:

$$\frac{d}{dx}(fg) = \frac{df}{dx}g + f\frac{dg}{dx}$$

$$\frac{d}{dx}\big(f(g(x))\big) = \frac{df}{dx}(g(x))\frac{dg}{dx}(x)$$

Computer programming and mathematics

Ex. Factorial $n! = 1.2..... .n$

If $n = 0$ then

$n! = 1$

else

$n! = n\,.\,(n-1)!$

Ex. Functions:

$$f(x) = x^2 + 2x + 1$$

Computer programming for mathematicians

“Smart” calculator (computing for mathematics)

```
1  x = sympy.Symbol('x')
2  expr = x**2*ln(x) - exp(sin(x))
3  expr
```

✓ 0.0s

$$x^2 \log(x) - e^{\sin(x)}$$

```
1  expr.diff(x)
```

✓ 0.0s

$$2x \log(x) - e^{\sin(x)} \cos(x) + x$$

```
1  theta = Symbol('θ')
2  expr = sin(theta)**2
3  expr
```

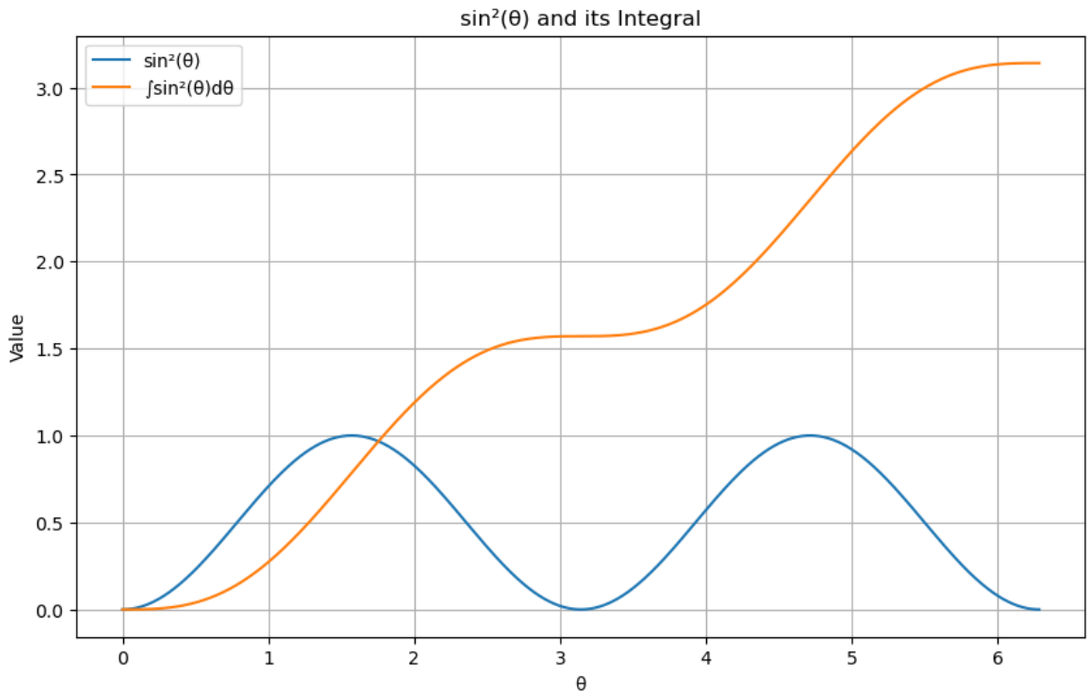
✓ 0.0s

$$\sin^2(\theta)$$

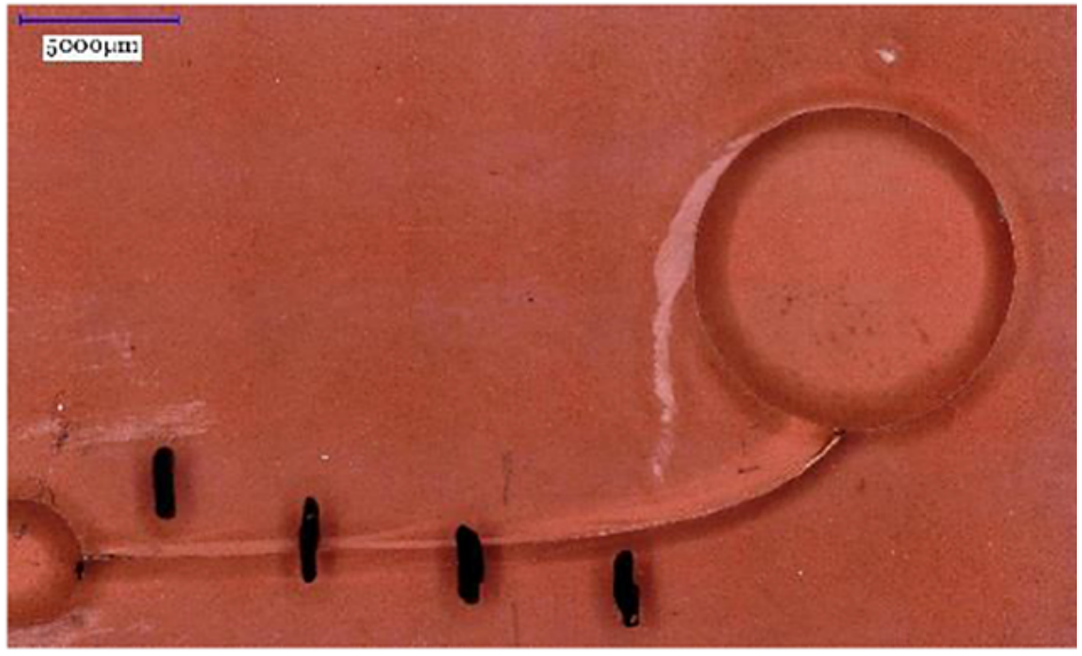
```
1  integral = integrate(expr, theta)
2  integral
```

✓ 0.0s

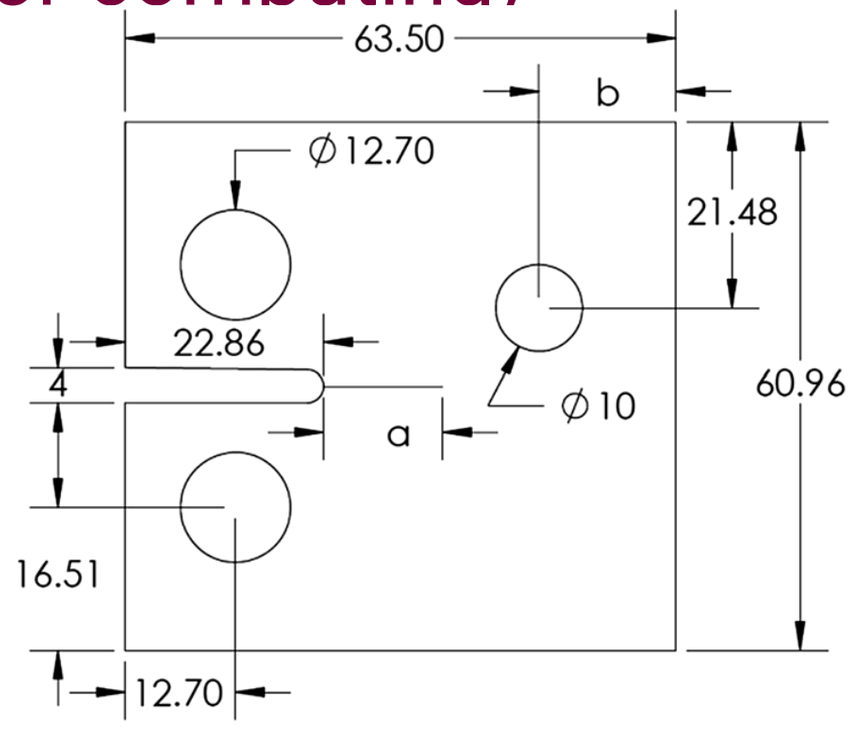
$$-\frac{\sin(\theta)\cos(\theta)}{2} + \frac{\theta}{2}$$

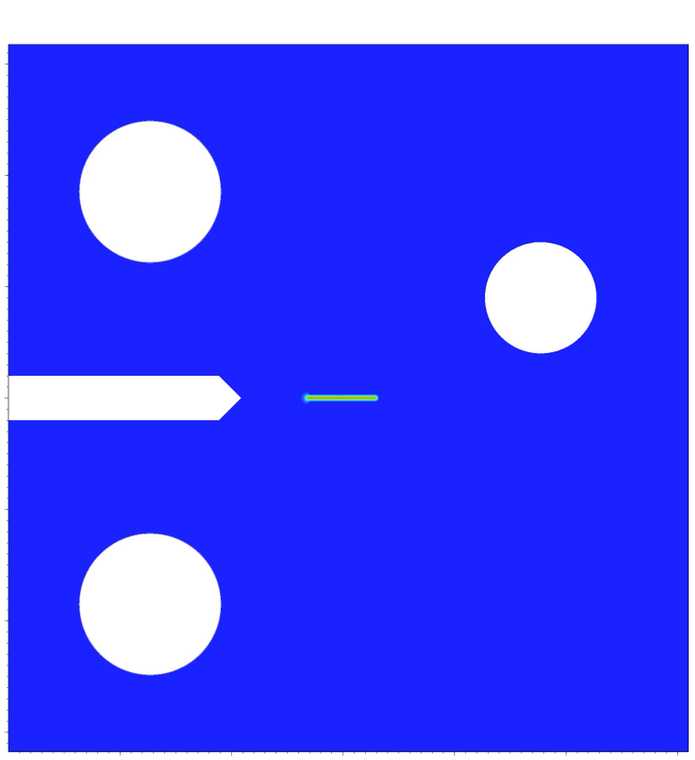


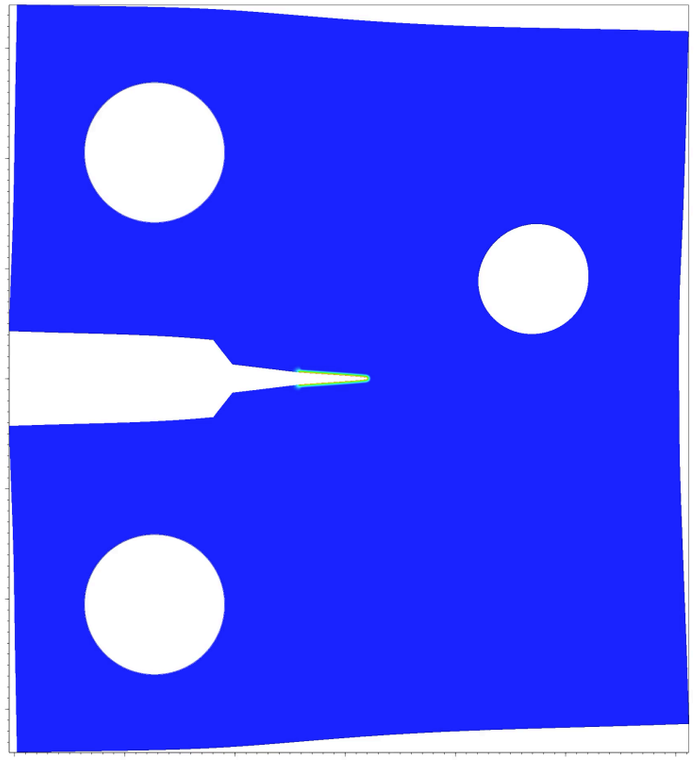
Scientific computing (mathematics for computing)



Pham Ravi-Chandar *IJF*, 2016







Course content

Chapters 1–6 + some of 8–10 from the free open-source textbook "Introduction to python programming" from openstax.

Specific tools for mathematics:

- numpy (basic math tools)
- matplotlib (plotting / graphing)
- pandas (managing data)
- scipy (scientific computations)
- sympy (symbolic computations)

Course evaluations

- Homework assignments after each chapter
- 2 mid-term exams
  February 3, 7:00 - 9:00 PM in PGCLL/B138.
  March 12, 7:00 - 9:00 PM in KTH/B135.
- 1 final exam

The final grade will be calculated using the maximum of grading schemes I and II as follows:

Assessment	Grading scheme I	Grading scheme II
Homework assigments	30% (drop one)	30% (drop one)
Two mid-term exams	40% (20% each)	20% (drop one)
Final exam	30%	50%

If a MSAF is filed for an assignment or a mid-term, they will be treated as the lowest (dropped) mark.

UDL and SAS accommodation

- You are automatically given a 72h grace period on all assignments. This satisfies the SAS accommodation “"Extensions on Assignments/Lab Reports”. Late submissions beyond this grace period will not be accepted.
- Exams will be 1h20, but the room will be booked for 2h, which will satisfy the SAS accommodation "Extra 30 Minutes per Hour for Quizzes, Tests and Exams”. Unless students have special accommodations for alternate times or small rooms, they are to write their tests with the rest of the class.
- Python notebook templates will be provided ahead of classes, and updated at the end of each section, which satisfies the "Access to PowerPoints/ Slides” accommodation.

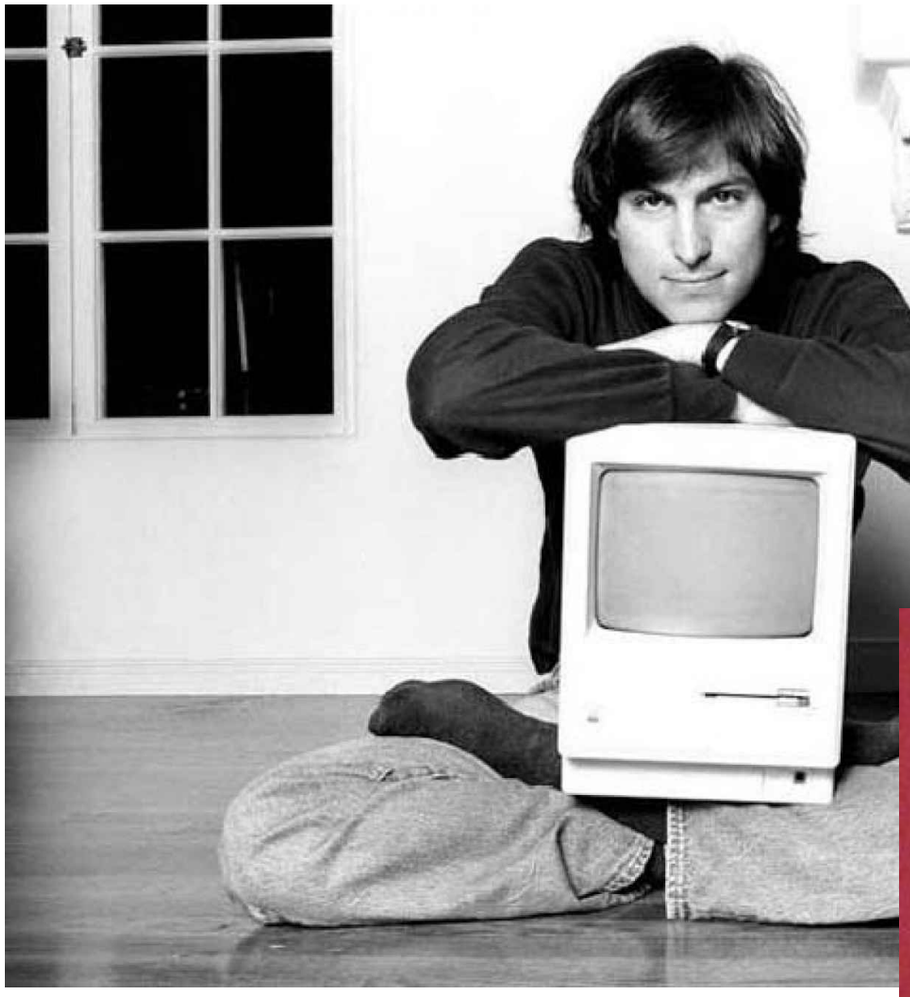
- Midterms and final exams will be computer based. Students are expected to bring a laptop capable of connecting to the university wifi, connect to https://mcmaster.syzygy.ca, and run on battery for up to 2h.

Gen-AI policy

Use of Gen-AI is **not allowed** for **any** assignment.

- Exception: assistive devices, spellcheckers, language tools.
- We reserve the right to ask students to go through their code line by line and explain it if we suspect that gen-AI has been used, or that students collaborated on assignments.

A short history of computer programming



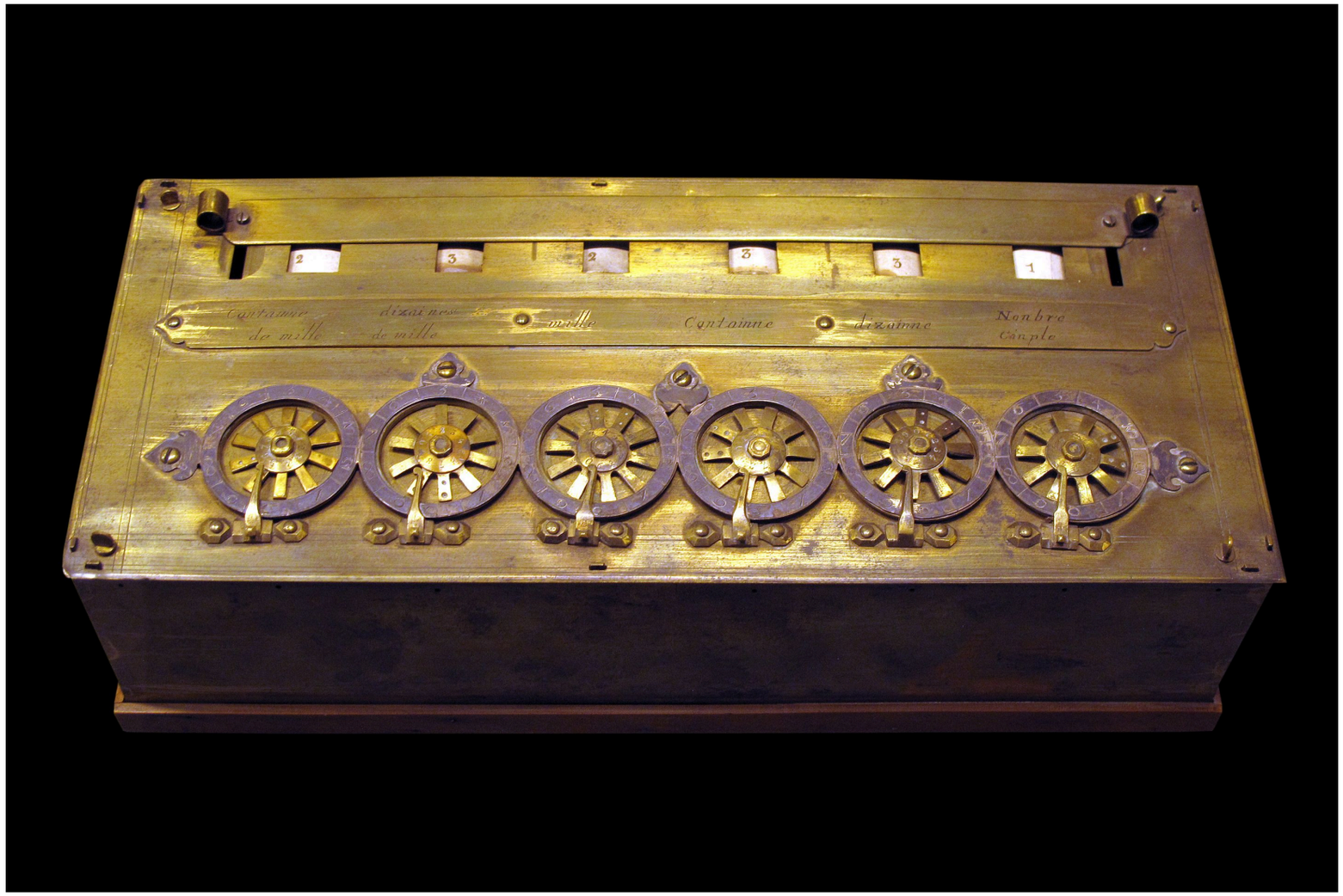
Steve Jobs

Mark Zuckerberg

Bill Gates

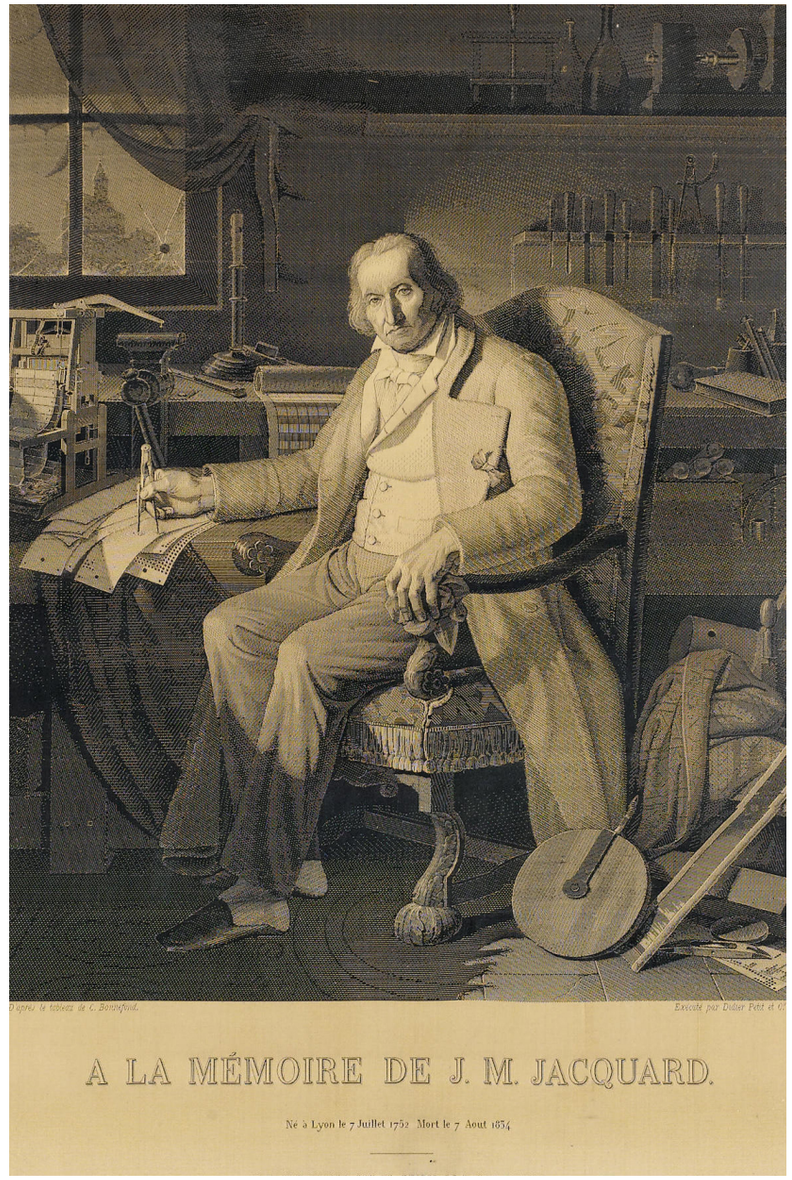
Blaise Pascal (1623 - 1662)

First mechanical calculator: the “Pascaline”



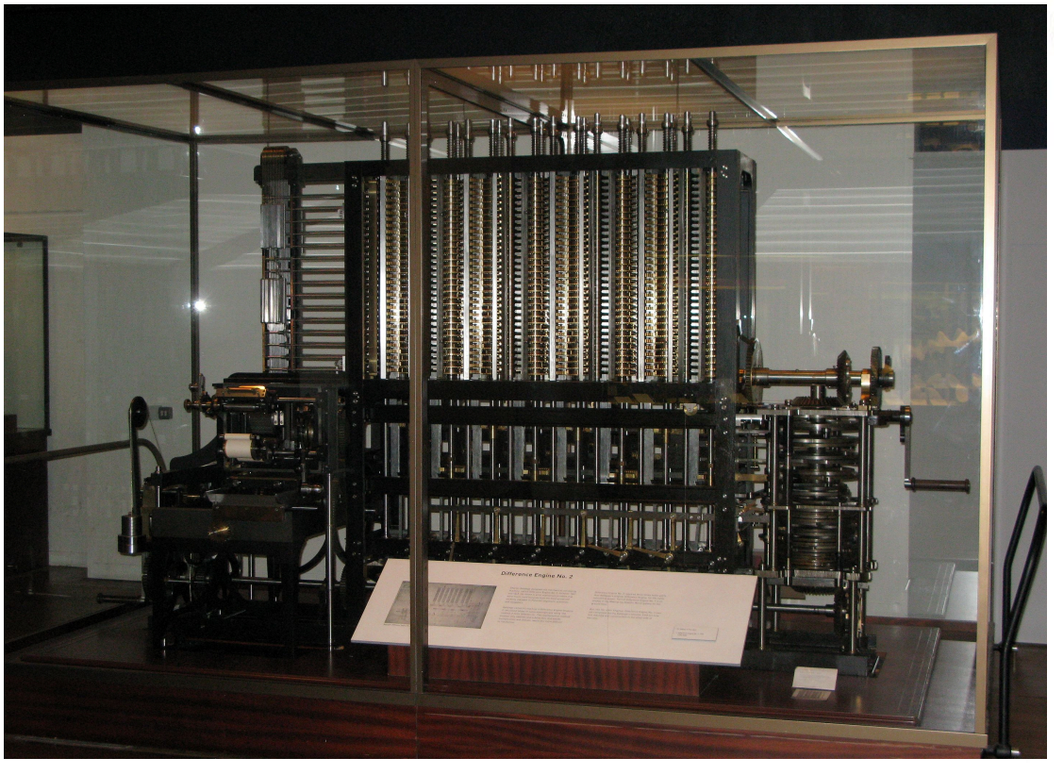
Joseph-Marie Jacquard (1752 - 1834)

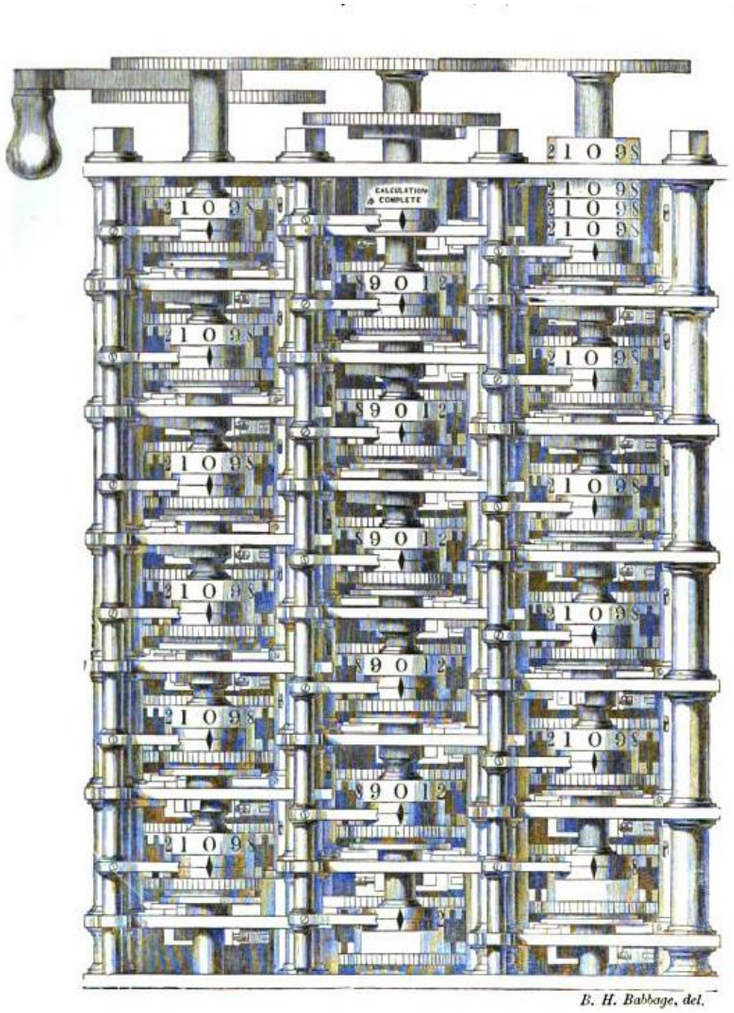
First programmable machine: the “Jacquard loom”

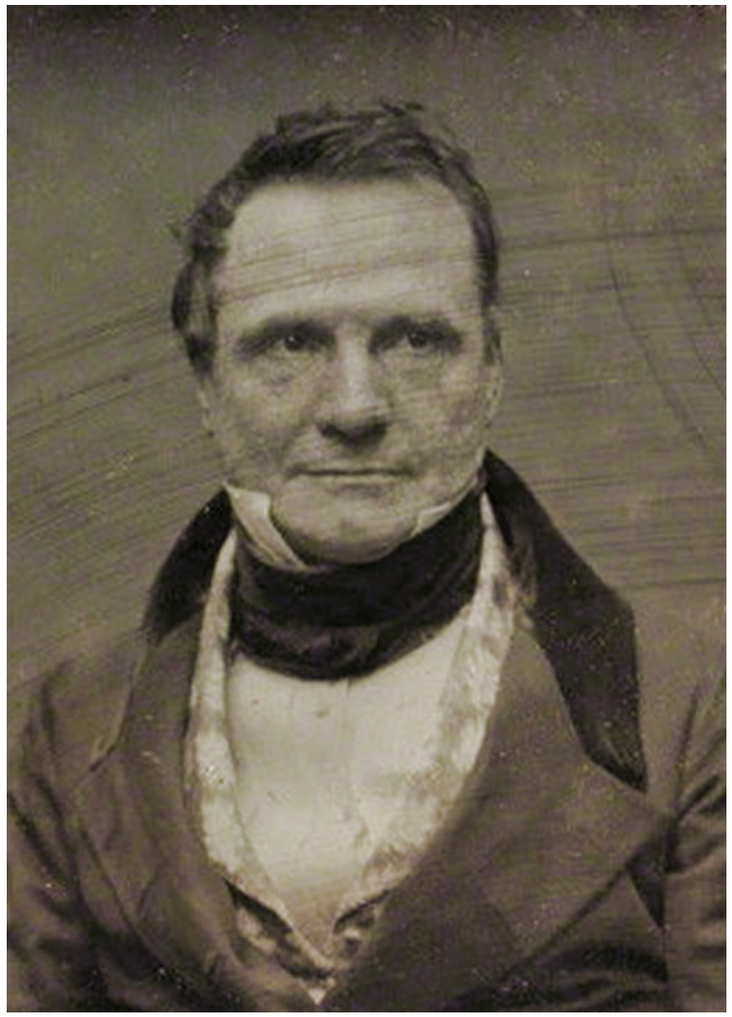


Charles Babbage (1781 - 1871)

First programmable computer: the “analytical engine”







Ada Lovelace (1815 - 1852)

First published computer program: “Note G”

Diagram for the computation by the Engine of the Numbers of Bernoulli. See Note G. (page 722 *et seq.*)

Number of Operation.	Nature of Operation.	Variables acted upon.	Variables receiving results.	Indication of change in the value on any Variable.	Statement of Results.	Data.			Working Variables.								
						$^1V_1$ 0 0 0 1	$^1V_2$ 0 0 0 2	$^1V_3$ 0 0 0 4	$^0V_4$ 0 0 0 0	$^0V_5$ 0 0 0 0	$^0V_6$ 0 0 0 0	$^0V_7$ 0 0 0 0	$^0V_8$ 0 0 0 0	$^0V_9$ 0 0 0 0	$^0V_{10}$ 0 0 0 0	$^0V_{11}$ 0 0 0 0	$^0V_{12}$ 0 0 0 0
						1	2	n									
1	×	$^1V_2 \times ^1V_3$	$^1V_4, ^1V_5, ^1V_6$	$\left\{\begin{matrix}^1V_2=^1V_2\\ ^1V_3=^1V_3\end{matrix}\right\}$	$=2n$	...	2	n	$2n$	$2n$	$2n$						
2	−	$^1V_4 - ^1V_1$	$^2V_4$	$\left\{\begin{matrix}^1V_4=^2V_4\\ ^1V_1=^1V_1\end{matrix}\right\}$	$=2n-1$	1	...	...	$2n-1$								
3	+	$^1V_5 + ^1V_1$	$^2V_5$	$\left\{\begin{matrix}^1V_5=^2V_5\\ ^1V_1=^1V_1\end{matrix}\right\}$	$=2n+1$	1	...	...	...	$2n+1$							
4	÷	$^2V_5 \div ^2V_4$	$^1V_{11}$	$\left\{\begin{matrix}^2V_5=^0V_5\\ ^2V_4=^0V_4\end{matrix}\right\}$	$=\frac{2n-1}{2n+1}$	...	...	...	0	0	...	...	...	...	...	$\frac{2n-1}{2n+1}$	
5	÷	$^1V_{11} \div ^1V_2$	$^2V_{11}$	$\left\{\begin{matrix}^1V_{11}=^2V_{11}\\ ^1V_2=^1V_2\end{matrix}\right\}$	$=\frac{1}{2}\cdot\frac{2n-1}{2n+1}$	...	2	...	...	...	...	...	...	...	...	$\frac{1}{2}\cdot\frac{2n-1}{2n+1}$	
6	−	$^0V_{13} - ^2V_{11}$	$^1V_{13}$	$\left\{\begin{matrix}^2V_{11}=^0V_{11}\\ ^0V_{13}=^1V_{13}\end{matrix}\right\}$	$=-\frac{1}{2}\cdot\frac{2n-1}{2n+1}=A_0$	...	...	...	...	...	...	...	...	...	...	0	.....
7	−	$^1V_3 - ^1V_1$	$^1V_{10}$	$\left\{\begin{matrix}^1V_3=^1V_3\\ ^1V_1=^1V_1\end{matrix}\right\}$	$=n-1\,(=3)$	1	...	n	...	...	...	...	...	...	$n-1$		
8	+	$^1V_2 + ^0V_7$	$^1V_7$	$\left\{\begin{matrix}^1V_2=^1V_2\\ ^0V_7=^1V_7\end{matrix}\right\}$	$=2+0=2$	...	2	...	...	...	...	2					
9	÷	$^1V_6 \div ^1V_7$	$^3V_{11}$	$\left\{\begin{matrix}^1V_6=^1V_6\\ ^0V_{11}=^3V_{11}\end{matrix}\right\}$	$=\frac{2n}{2}=A_1$	...	...	...	...	...	$2n$	2	...	...	...	$\frac{2n}{2}=A_1$	
10	×	$^1V_{21} \times ^3V_{11}$	$^1V_{12}$	$\left\{\begin{matrix}^1V_{21}=^1V_{21}\\ ^3V_{11}=^3V_{11}\end{matrix}\right\}$	$=B_1\cdot\frac{2n}{2}=B_1A_1$	...	...	...	...	...	...	...	...	...	...	$\frac{2n}{2}=A_1$	$B_1\cdot\frac{2n}{2}=B_1A_1$
11	+	$^1V_{12} + ^1V_{13}$	$^2V_{13}$	$\left\{\begin{matrix}^1V_{12}=^0V_{12}\\ ^1V_{13}=^2V_{13}\end{matrix}\right\}$	$=-\frac{1}{2}\cdot\frac{2n-1}{2n+1}+B_1\cdot\frac{2n}{2}$	...	...	...	...	...	...	...	...	...	...	.....	0
12	−	$^1V_{10} - ^1V_1$	$^2V_{10}$	$\left\{\begin{matrix}^1V_{10}=^2V_{10}\\ ^1V_1=^1V_1\end{matrix}\right\}$	$=n-2\,(=2)$	1	...	...	...	...	...	...	...	...	$n-2$		
13	−	$^1V_6 - ^1V_1$	$^2V_6$	$\left\{\begin{matrix}^1V_6=^2V_6\\ ^1V_1=^1V_1\end{matrix}\right\}$	$=2n-1$	1	...	...	...	...	$2n-1$						
14	+	$^1V_1 + ^1V_7$	$^2V_7$	$\left\{\begin{matrix}^1V_1=^1V_1\\ ^1V_7=^2V_7\end{matrix}\right\}$	$=2+1=3$	1	...	...	...	...	...	3					
15	÷	$^2V_6 \div ^2V_7$	$^1V_8$	$\left\{\begin{matrix}^2V_6=^2V_6\\ ^2V_7=^2V_7\end{matrix}\right\}$	$=\frac{2n-1}{3}$	...	...	...	...	...	$2n-1$	3	$\frac{2n-1}{3}$				
16	×	$^1V_8 \times ^3V_{11}$	$^4V_{11}$	$\left\{\begin{matrix}^1V_8=^0V_8\\ ^3V_{11}=^4V_{11}\end{matrix}\right\}$	$=\frac{2n}{2}\cdot\frac{2n-1}{3}$	...	...	...	...	...	...	...	0	...	...	$\frac{2n}{2}\cdot\frac{2n-1}{3}$	
17	−	$^2V_6 - ^1V_1$	$^3V_6$	$\left\{\begin{matrix}^2V_6=^3V_6\\ ^1V_1=^1V_1\end{matrix}\right\}$	$=2n-2$	1	...	...	...	...	$2n-2$						
18	+	$^1V_1 + ^2V_7$	$^3V_7$	$\left\{\begin{matrix}^2V_7=^3V_7\\ ^1V_1=^1V_1\end{matrix}\right\}$	$=3+1=4$	1	...	...	...	...	...	4					
19	÷	$^3V_6 \div ^3V_7$	$^1V_9$	$\left\{\begin{matrix}^3V_6=^3V_6\\ ^3V_7=^3V_7\end{matrix}\right\}$	$=\frac{2n-2}{4}$	...	...	...	...	...	$2n-2$	4	...	$\frac{2n-2}{4}$	...	$\left\{\frac{2n}{2}\cdot\frac{2n-1}{3}\cdot\frac{2n-2}{3}\right\}$	
20	×	$^1V_9 \times ^4V_{11}$	$^5V_{11}$	$\left\{\begin{matrix}^1V_9=^0V_9\\ ^4V_{11}=^5V_{11}\end{matrix}\right\}$	$=\frac{2n}{2}\cdot\frac{2n-1}{3}\cdot\frac{2n-2}{4}=A_3$	...	...	...	...	...	...	...	...	0		$=A_3$	
21	×	$^1V_{22} \times ^5V_{11}$	$^0V_{12}$	$\left\{\begin{matrix}^1V_{22}=^1V_{22}\\ ^0V_{12}=^2V_{12}\end{matrix}\right\}$	$=B_3\cdot\frac{2n}{2}\cdot\frac{2n-1}{3}\cdot\frac{2n-2}{3}=B_3A_3$	...	...	...	...	...	...	...	...	...	...	0	B_3A_3
22	+	$^2V_{12} + ^2V_{13}$	$^3V_{13}$	$\left\{\begin{matrix}^2V_{12}=^0V_{12}\\ ^2V_{13}=^3V_{13}\end{matrix}\right\}$	$=A_0+B_1A_1+B_3A_3$	...	...	...	...	...	...	...	...	...	...	.....	0
23	−	$^2V_{10} - ^1V_1$	$^3V_{10}$	$\left\{\begin{matrix}^2V_{10}=^3V_{10}\\ ^1V_1=^1V_1\end{matrix}\right\}$	$=n-3\,(=1)$	1	...	...	...	...	...	...	...	...	$n-3$		

Here follows a repetition of Operations thirteen to twenty-three.

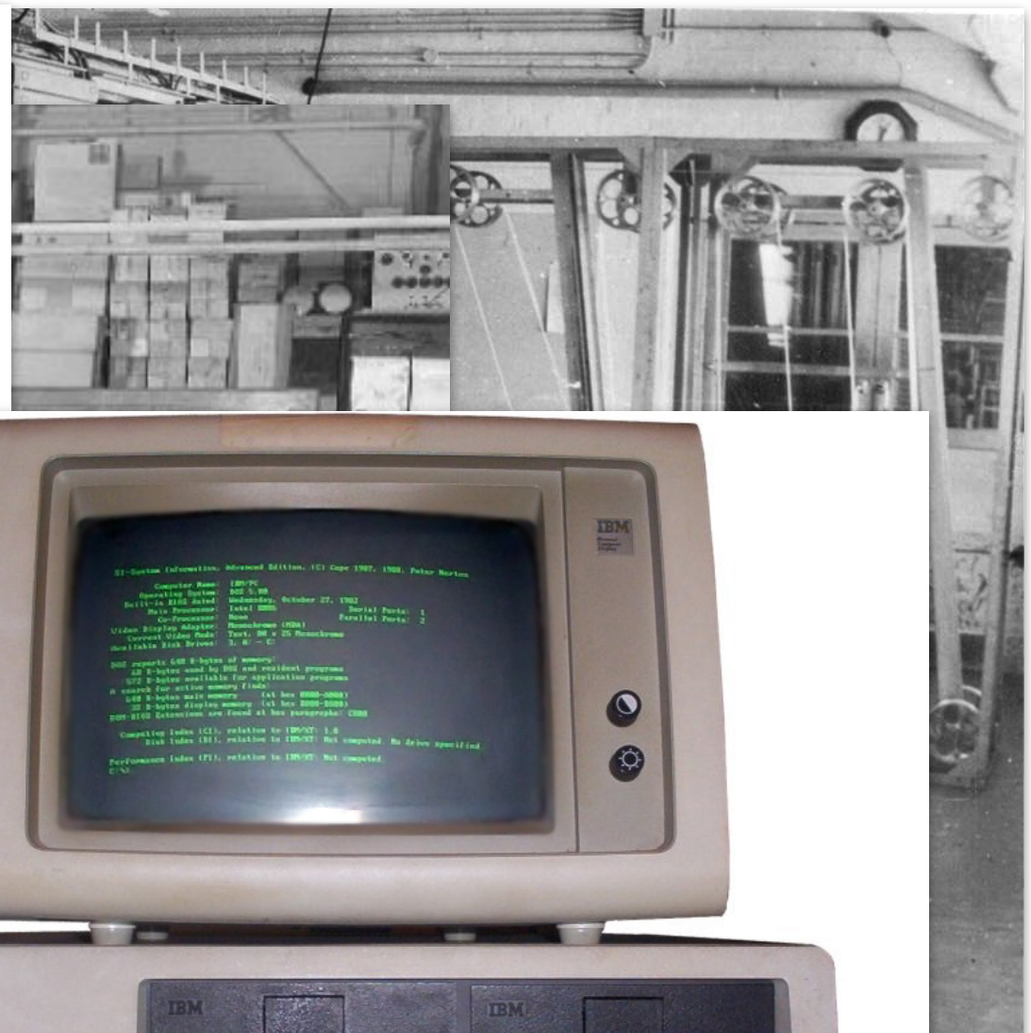
24	+	$^4V_{13} + ^0V_{24}$	$^1V_{24}$	$\left\{\begin{matrix}^4V_{13}=^0V_{13}\\ ^0V_{24}=^1V_{24}\end{matrix}\right\}$	$=B_7$	...	...	...	...	...	...	...	...	...	...	.....	.....
25	+	$^1V_1 + ^1V_3$	$^1V_3$	$\left\{\begin{matrix}^1V_1=^1V_1\\ ^1V_3=^1V_3\\ ^5V_6=^0V_6\\ ^5V_7=^0V_7\end{matrix}\right\}$	$=n+1=4+1=5$ by a Variable-card. by a Variable card.	1	...	$n+1$	...	...	0	0					

Early “modern" computers

Colossus 10, Bletchey park, 1943-45, UK

Apollo guidanc

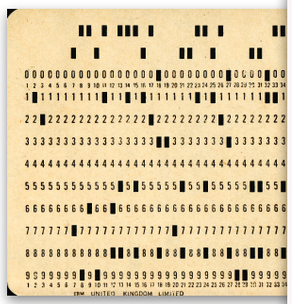
Imaged by Heritage Auctions, HA.com



SA

IBM 5150 PC, 1980

Levels of programming language

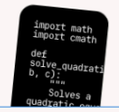


B Write a code that solves a quadratic equation

Edit

I'll create a Python function that solves quadratic equations using the quadratic formula. This will handle all cases including real and complex solutions.

Quadratic Equation Solver
Code



This code provides a comprehensive quadratic equation solver that:

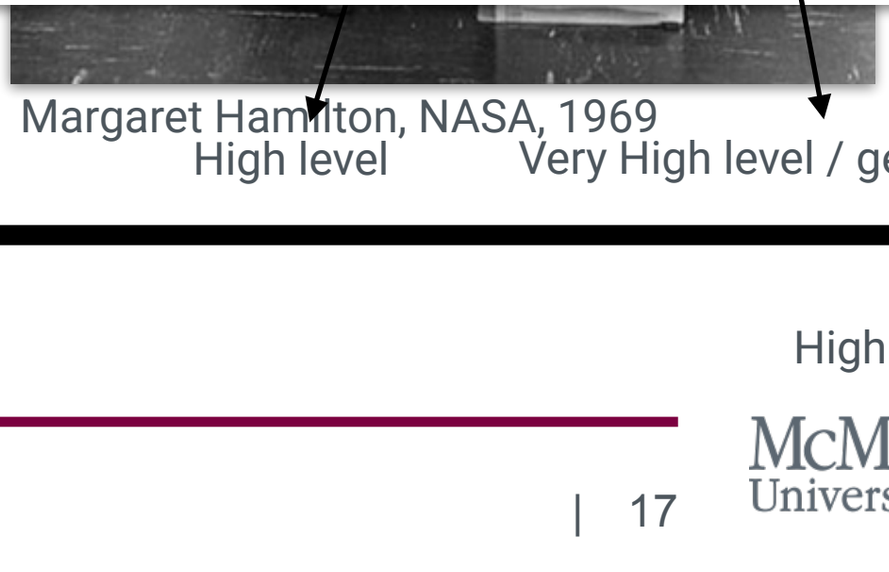
Key Features:

- Handles all types of quadratic equations ($ax^2 + bx + c = 0$)
- Calculates the discriminant to determine the nature of solutions
- Returns real solutions when discriminant ≥ 0
- Returns complex solutions when discriminant < 0
- Handles edge cases (when a = 0, making it linear)
- Includes formatted output for easy reading

How it works:

1. Uses the quadratic formula: $x = (-b \pm \sqrt{(b^2-4ac)}) / (2a)$
2. Calculates discriminant (b^2-4ac) to determine solution types:
   - Positive: two real solutions
   - Zero: one real solution (repeated root)
   - Negative: two complex solutions

```python
import math
import cmath

def solve_quadratic(a, b, c):
    """
    Solves a quadratic equation of the form ax² + bx + c = 0

    Args:
        a (float): coefficient of x²
        b (float): coefficient of x
        c (float): constant term

    Returns:
        tuple: (solution1, solution2, discriminant)
    """

    # Check if it's actually a quadratic equation
    if a == 0:
        if b == 0:
            if c == 0:
                return "Infinite solutions", "Infinite solutions", Non
            else:
                return "No solution", "No solution", None
        else:
            # Linear equation: bx + c = 0
            solution = -c / b
            return solution, solution, None

    # Calculate discriminant
    discriminant = b**2 - 4*a*c
```



Margaret Hamilton, NASA, 1969

Machine code

Low level

High level

Very High level / genAI

Low

High

Jupyter notebook interface

Go to avenue, click on the link in the first announcement