Group Members: Harsh Dixit and Aarnav Subbaramu
Group Name: BhaiTech Systems
Comp312
17 Jan 2025

Final Project Part- II

Q) Use the tools and techniques discussed in the class to compare the memory footprint and cache behaviors of those two programs.

1. Memory Bound Operations Analysis (from lecture page 6):

Euler Method:
- Higher memory consumption due to:
  - Creates and stores all derivatives in a list (`derivatives = []`)
  - Each iteration appends a new tuple to the list
  - Memory grows linearly with number of steps

RK4 Method:
- Lower memory consumption because:
  - No list storage, just variables k1, k2, k3, k4
  - Only stores current calculation values
  - Memory usage remains constant regardless of steps

2. CPU Bound Operations (from lecture page 6):

Euler Method:
- Uses list operations (append)
- Single exponential calculation per iteration
- Two derivative calculations per step

RK4 Method:
- More CPU intensive due to:
  - First derivative: 4 exponential calculations (k1-k4)
  - Second derivative: 3 exponential calculations (k1,k2,k4)
  - More arithmetic operations per step

3. Cache Behavior Analysis (from lecture page 11):

Euler Method:
- Poor cache utilization:
  - Growing list size may exceed cache
  - Non-contiguous memory access due to list appending
  - Each append operation may trigger cache misses

RK4 Method:
- Better cache utilization:
  - Fixed number of variables (k1-k4)
  - Reuses same memory locations
  - Sequential memory access pattern
  - Variables likely to stay in cache

4. Memory Usage Optimization (from lecture page 9):

Euler Method:
- Less optimized because:
  - Creates new objects repeatedly (tuples in list)
  - Uses unnecessary data structures (list for storage)
  - Memory grows with input size

RK4 Method:
- More optimized because:
  - Reuses objects (k1-k4 variables)
  - No unnecessary data structures
  - Constant memory usage

5. Loop Optimization (from lecture page 10):

Euler Method:
- Less optimized loops:
  - Extra operations for list management
  - Memory allocation in each iteration

RK4 Method:
- More optimized loops:
  - Direct calculations without storage
  - No memory allocation in loop

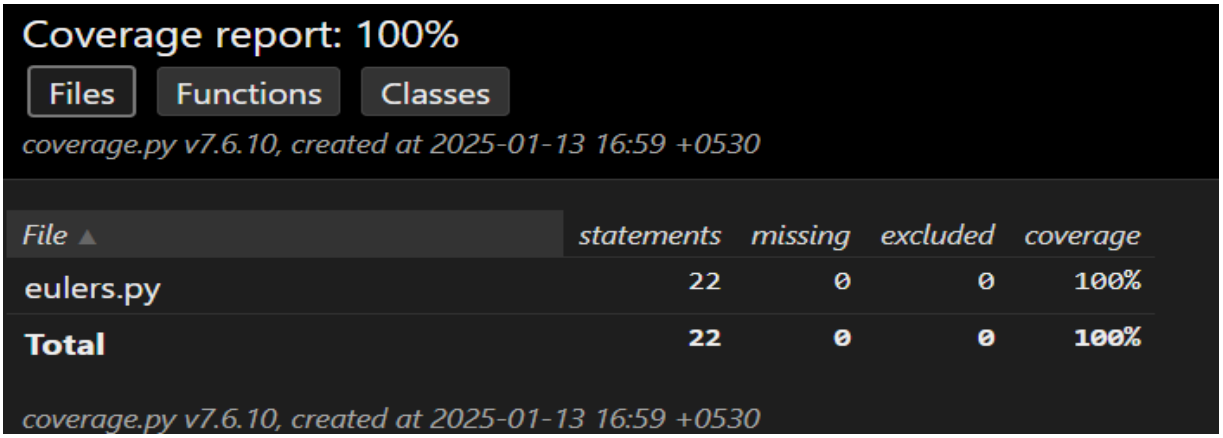
- Minimal loop overhead

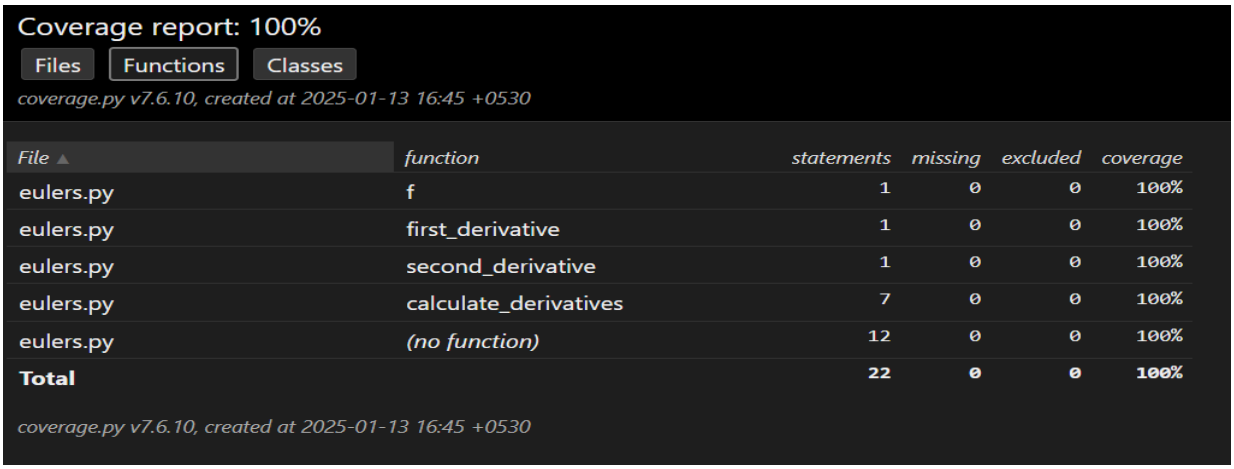
Essentially, the RK4 method is more optimized for memory footprint and cache behavior, though it uses more CPU resources. This aligns with the lecture's emphasis on memory optimization and cache efficiency over raw computation speed in many cases.

To analyse the test coverage of the implemented Python code, we utilized the *coverage* tool. The process began with installing the package using *pip install coverage*. Tests were executed with the command *coverage run -m pytest*, which monitored code execution during testing. Following this, coverage reports were generated using:

- *coverage report* for a detailed text summary,
- *coverage html* for an interactive HTML report.

Below are the screenshots of Coverage Reports.





Coverage report: 100%

Files Functions Classes

coverage.py v7.6.10, created at 2025-01-13 16:45 +0530

File ▲	class	statements	missing	excluded	coverage
eulers.py	*(no class)*	22	0	0	100%
Total		**22**	**0**	**0**	**100%**

coverage.py v7.6.10, created at 2025-01-13 16:45 +0530

Coverage report: 100%

Files Functions Classes

coverage.py v7.6.10, created at 2025-01-13 16:55 +0530

File ▲	statements	missing	excluded	coverage
RK-4.py	29	0	0	100%
Total	**29**	**0**	**0**	**100%**

coverage.py v7.6.10, created at 2025-01-13 16:55 +0530

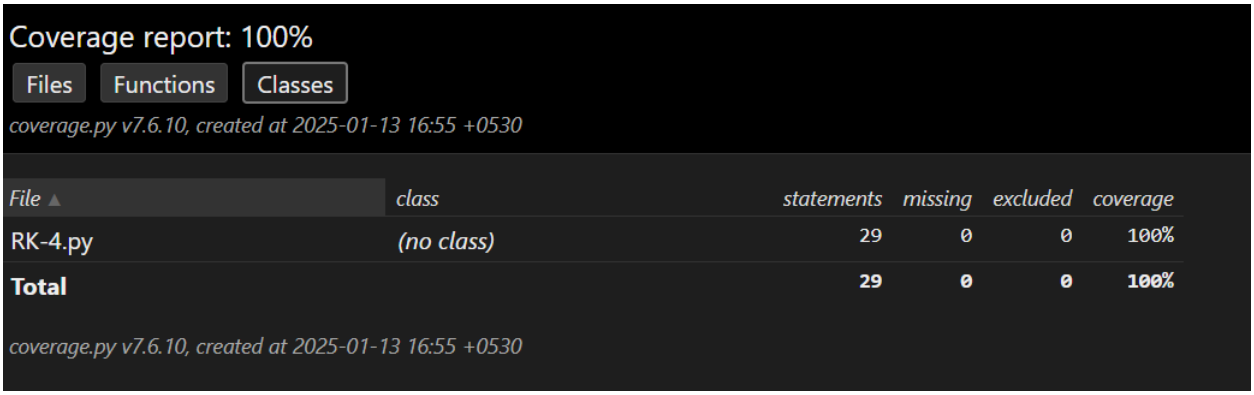
Coverage report: 100%

Files Functions Classes

coverage.py v7.6.10, created at 2025-01-13 16:55 +0530

File ▲	function	statements	missing	excluded	coverage
RK-4.py	f	2	0	0	100%
RK-4.py	first_derivative	5	0	0	100%
RK-4.py	second_derivative	4	0	0	100%
RK-4.py	main	12	0	0	100%
RK-4.py	*(no function)*	6	0	0	100%
Total		**29**	**0**	**0**	**100%**

coverage.py v7.6.10, created at 2025-01-13 16:55 +0530



Coverage report: 100%

Files Functions Classes

coverage.py v7.6.10, created at 2025-01-13 16:55 +0530

File ▲	class	statements	missing	excluded	coverage
RK-4.py	*(no class)*	29	0	0	100%
Total		**29**	**0**	**0**	**100%**

coverage.py v7.6.10, created at 2025-01-13 16:55 +0530