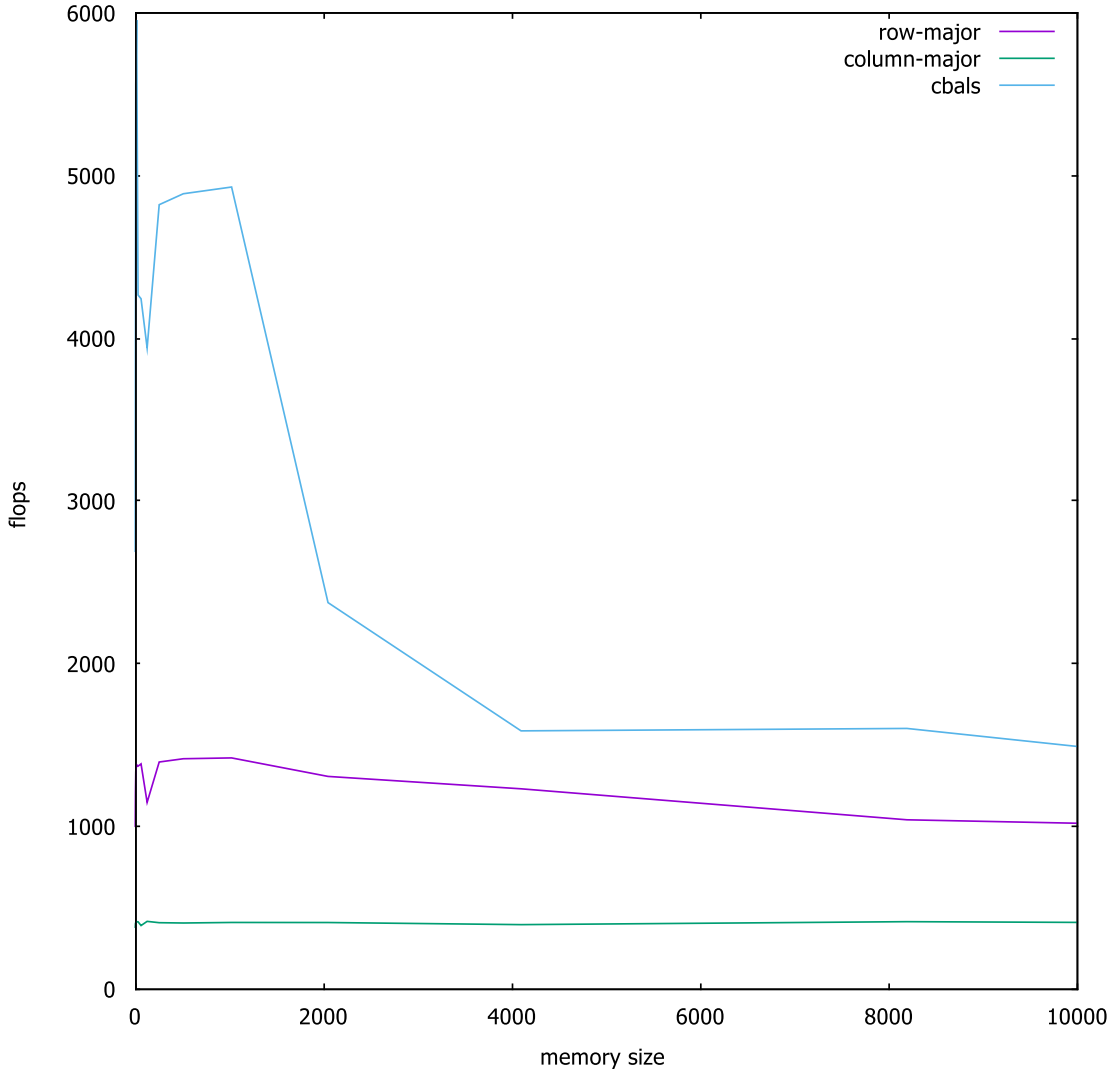
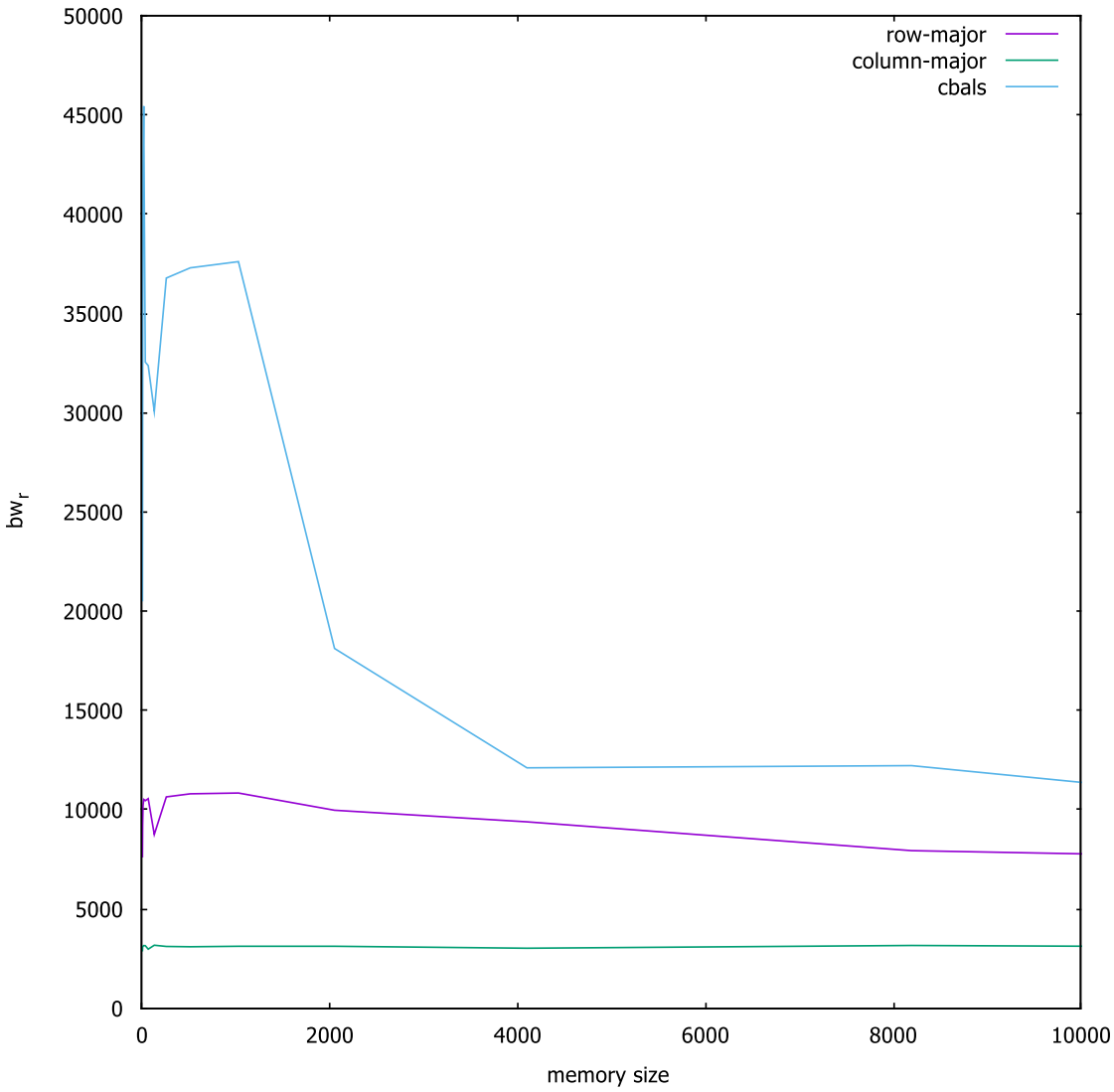
一、 以向量內積為標準的測試

- mem_size vs. flops/s

  memory size 大於 10000 後為平滑取線差異不大

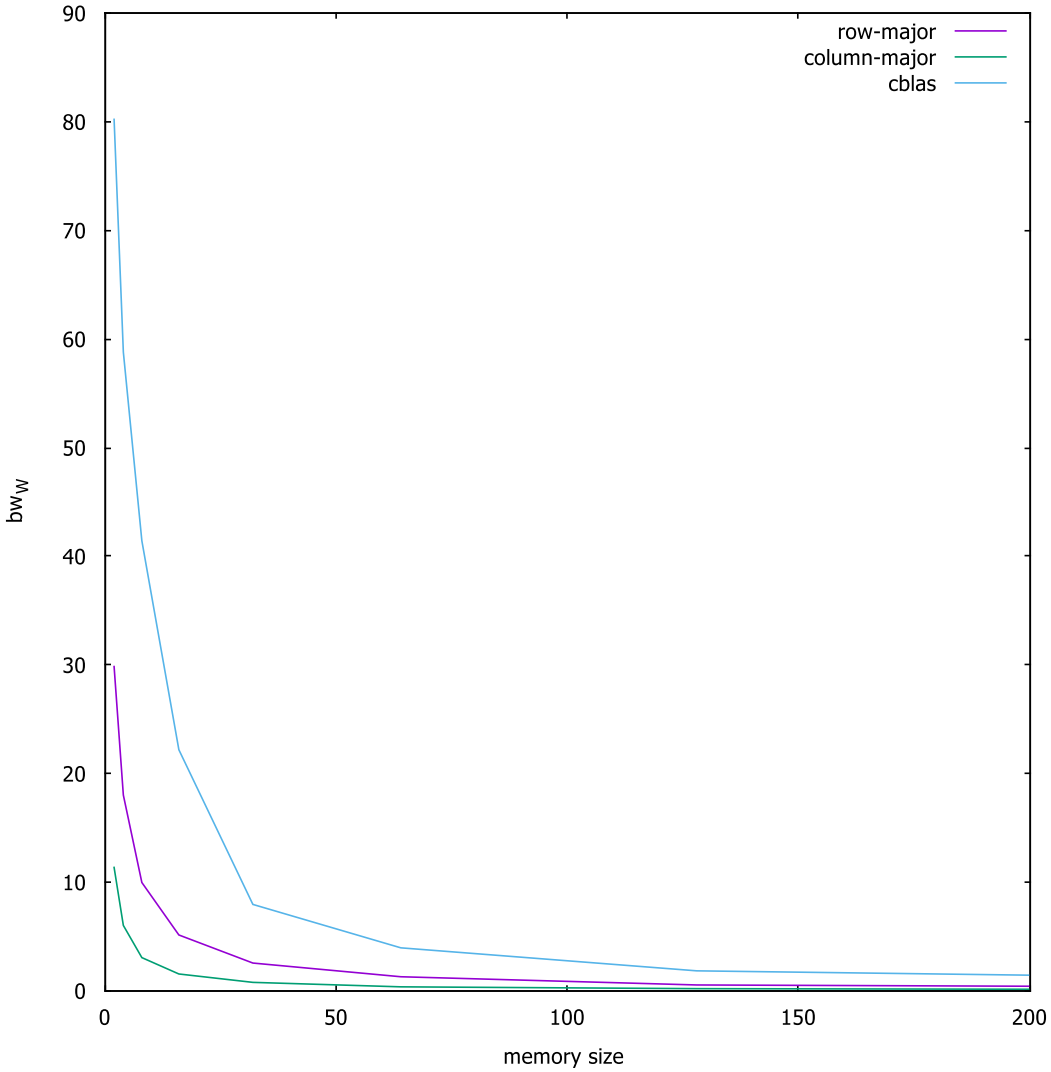


- mem_size vs. bw_r



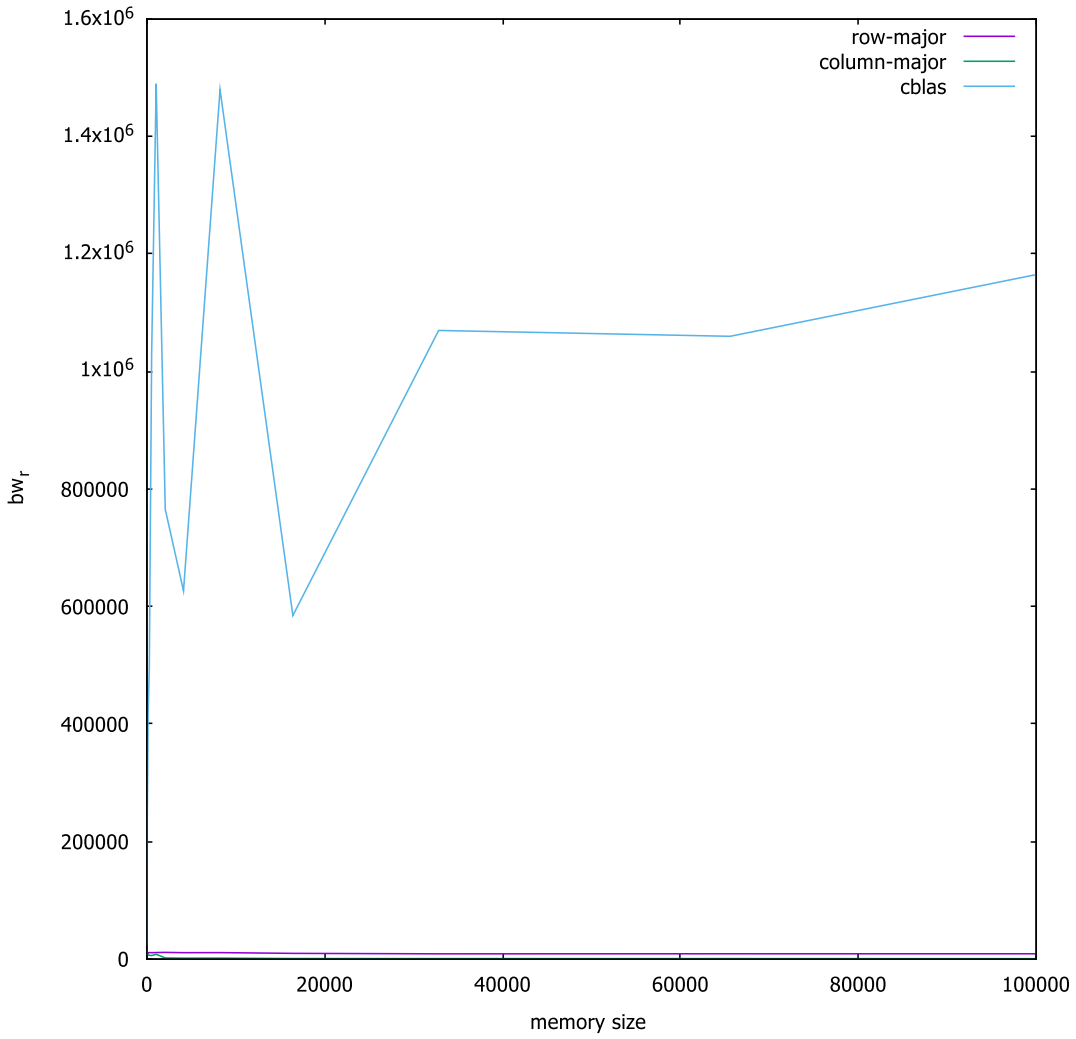
- mem_size vs. bw_w

  memory_size 超過 200KB 後頻寬幾乎接近 0

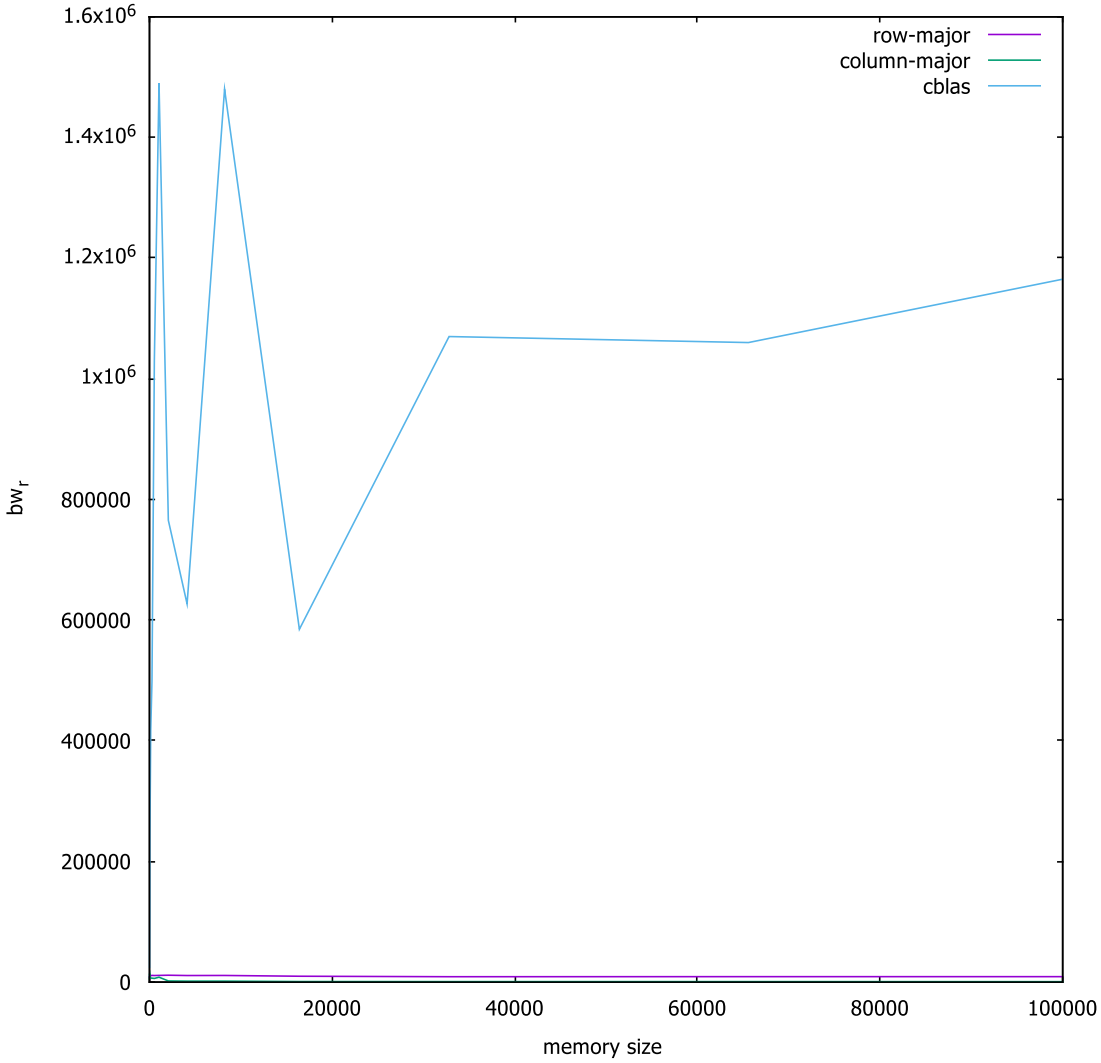


二、 以矩陣-向量相乘為標準的測試

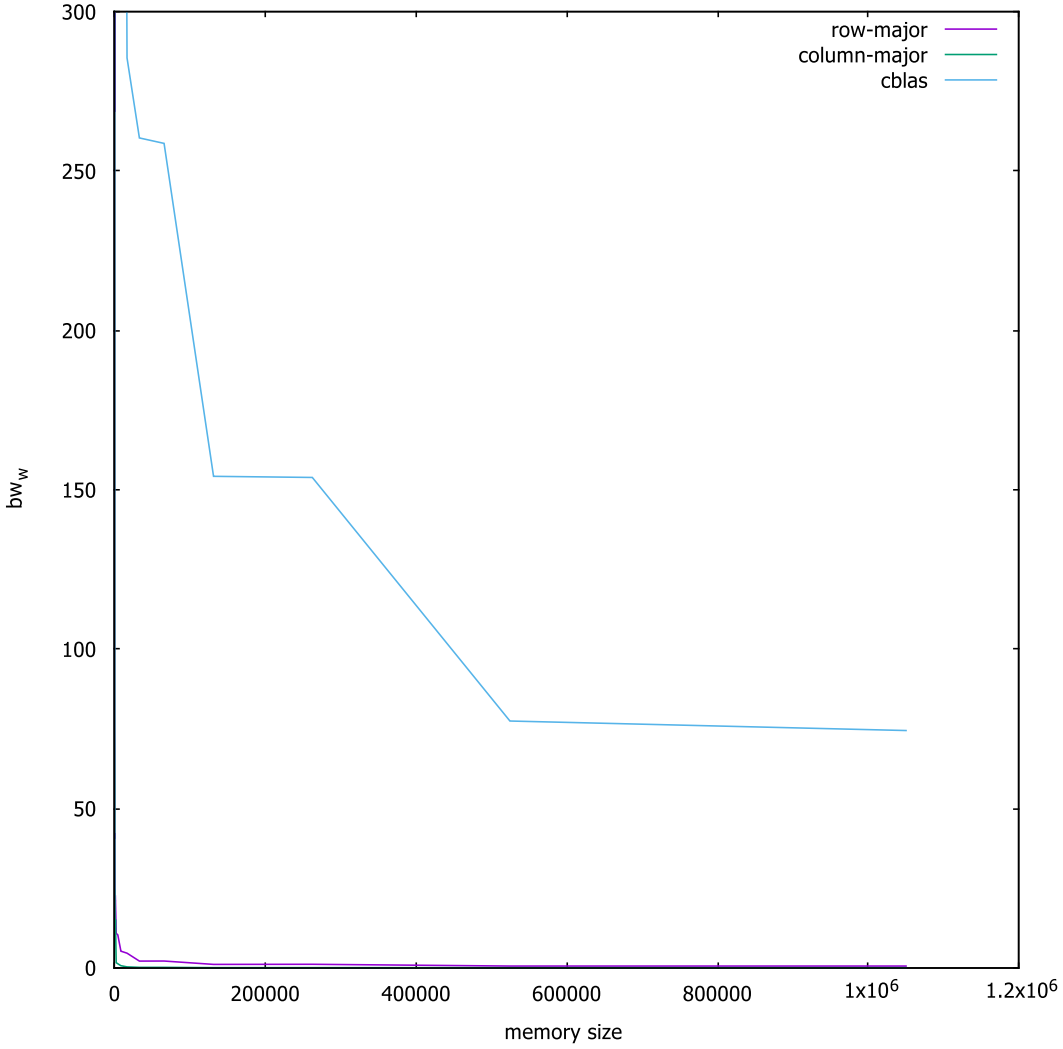
- mem_size vs. flops/s



- mem_size vs. bw_r

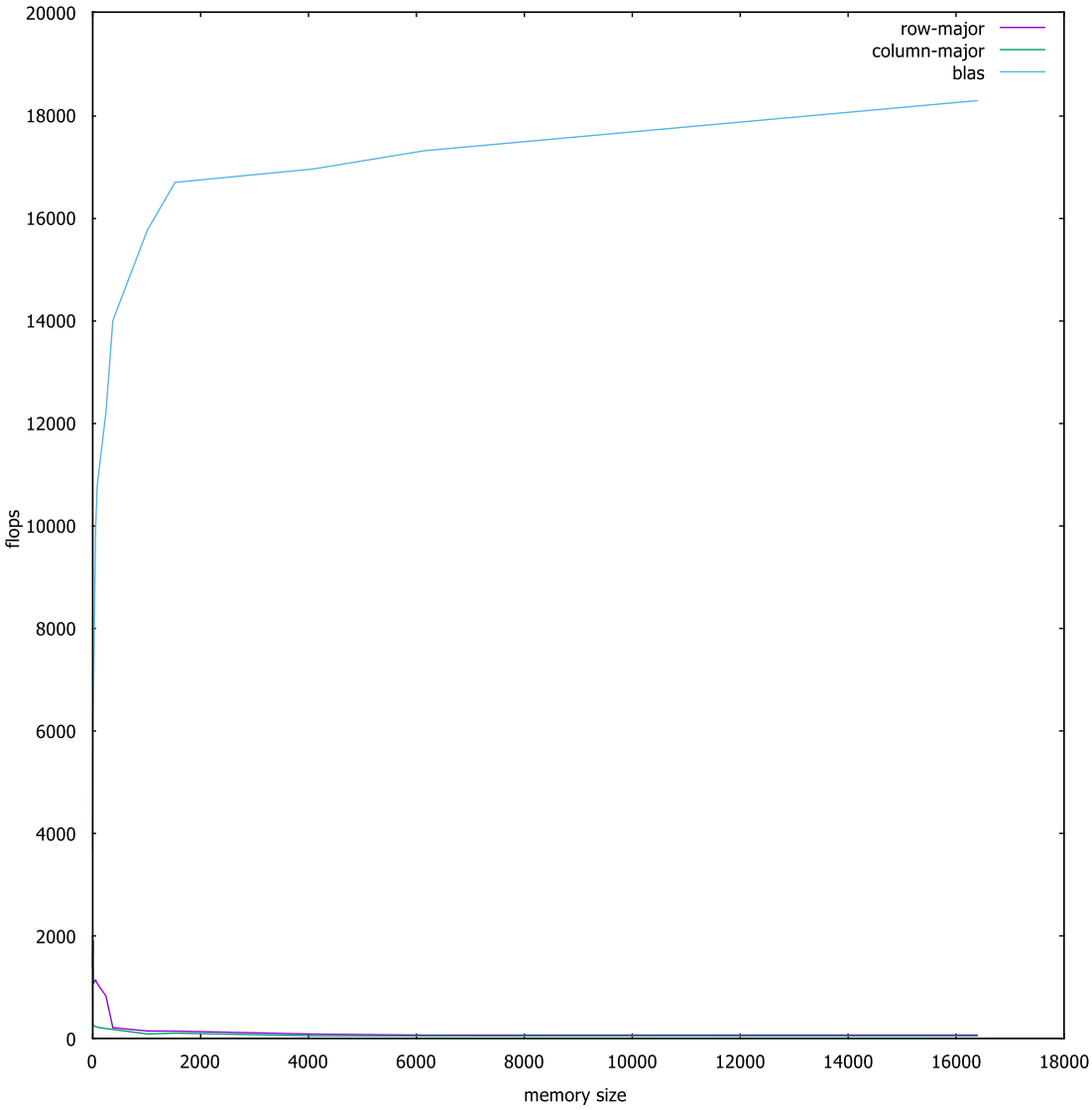


- mem_size vs. bw_w

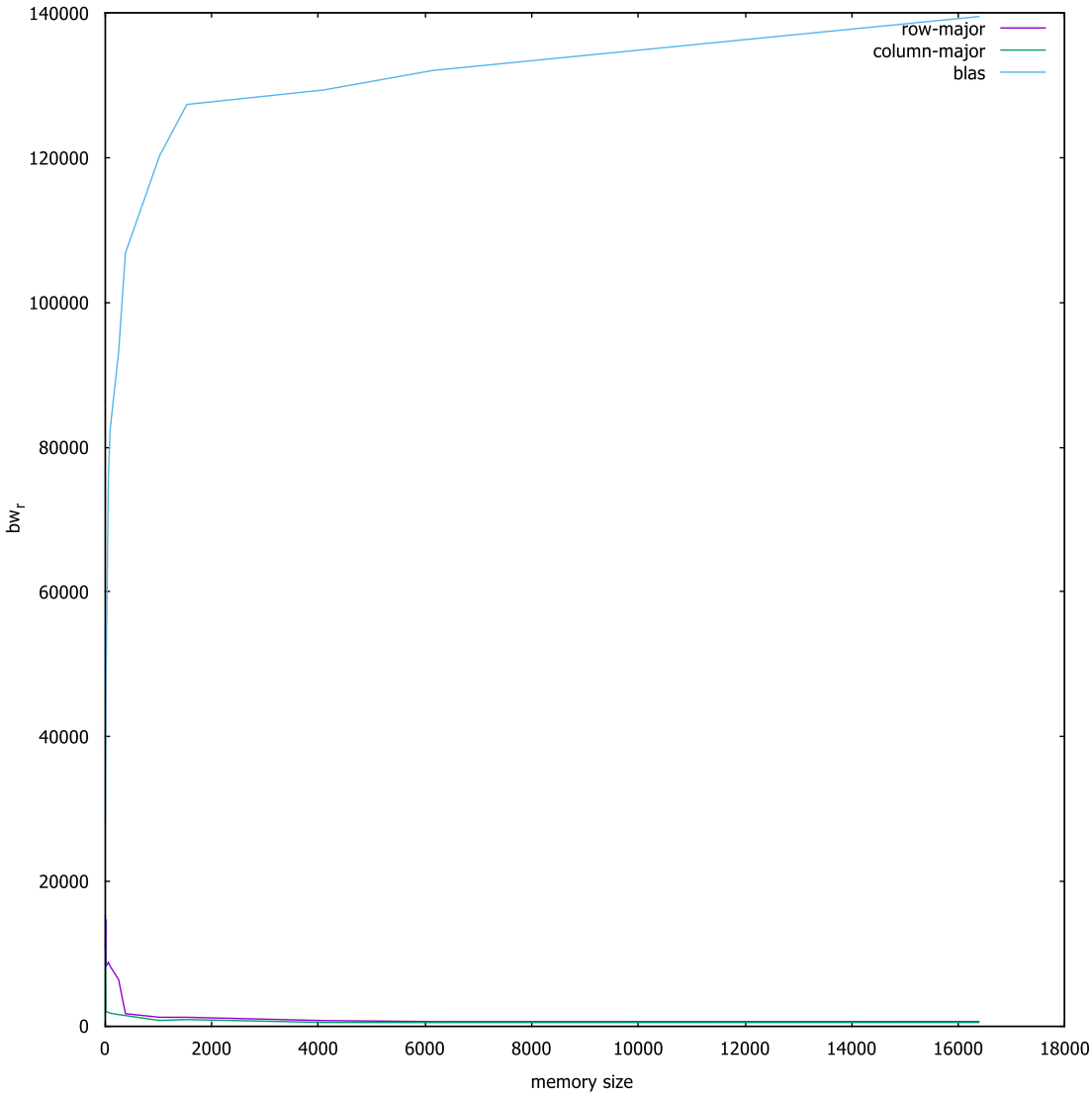


三、 以矩陣相乘為標準的測試

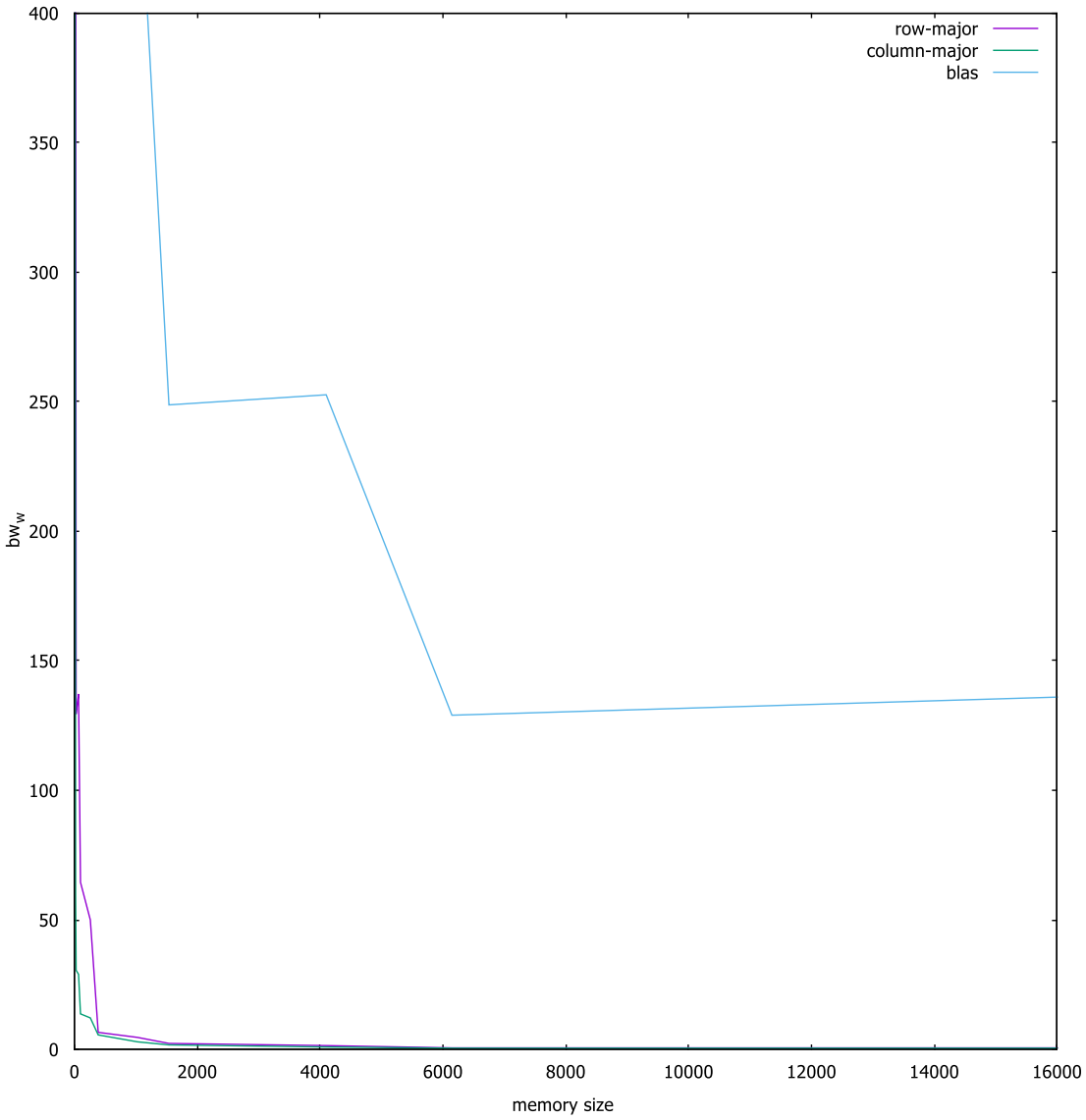
- mem_size vs. flops/s



- mem_size vs. bw_r



- mem_size vs. bw_w+



四、 結論

1. BLAS 加速遠高過作業一的兩種版本，但是計時是用 0.1 秒執行次數下去計算，發現 BLAS 在第一次執行的速度遠低於另外兩種版本，但是執行完第一次後的速度又大幅提升，可能是第一次計算是將數據放進 catch 中之後再從 catch 中拿出來計算。

2. BLAS 中向量內積的加速最為穩定，檔案大小達 100M 時與 10M 的矩陣計算速度相差不大。

3. BLAS 的效能表現呈現震盪(忽大忽小)，猜測可能是跟放入快取記憶體的數據剛好用完，運算速度驟降，下次放入後開始回升速度。

4. Row-major 符合 C 語言的記憶體讀寫模式，速度又比 column-major 快速許多
   屏除 blash 版本後的 br_w 比較，局部擷取示意圖