Quantum Information and Computing 2021-2022

3^{rd} week assignment
November 23, 2021

Campesan Giulia

Università degli Studi di Padova

Theory

Scaling of the matrix-matrix multiplication

- Given two matrices $A_{(M,K)}$, $B_{(K,N)}$ we can compute their product $C_{m,n} = \sum_k A_{m,k} B_{k,n}$
- the computation is achieved with 3 nested for loops: their order controls how the result matrix C is accessed and loaded. In particular, this will affect the program performance due to cache exploitation
- we consider square matrices with sizes in range [20, 2000] with step 20

Random matrix theory

Considering a random Hermitian matrix:

- we retrieve their spectrum $\{\lambda\}$ through the ZHEEV subroutine in the LAPACK library
- considering the eigenvalues in crescent order, we compute the normalized spacing $s_i = \frac{\Delta\lambda_i}{\langle \Delta\lambda \rangle}$, with $\Delta\lambda_i = \lambda_{i+1} - \lambda_i$
- we expect the distribution of the so-obtained $s = \{s_i\}$ to be well-approximated by the Wigner sumrise distribution $P_{th}(s) = \frac{32}{\pi^2} \cdot s^2 e^{-\frac{4}{\pi}s^2}$. Then, we perform a fit with the function $P(s) = a \cdot s^\alpha \cdot \exp\left(-b \cdot s^\beta\right)$ to retrieve the actual trend of our data.

Scaling of matrix-matrix multiplication

Università degli Studi di Padova

Code

The python script *execution.py* generates an array of matrix sizes and launches the compilation and execution of fortran program and libraries contained in *matrixmultiplication.f90 debugging.f90 performance.f90*

Results

In particular, the matrix size ranges from $N_{min} = 20$ to $N_{max} = 2000$ with a step of 20. Here is reported the trend of the CPUTIME $[s]$ vs matrix size for the two user-defined functions and the built-in MATMUL subroutine. Performing the multiplication by columns is faster and more stable, thanks to the exploitation of subsequent element in the cache.

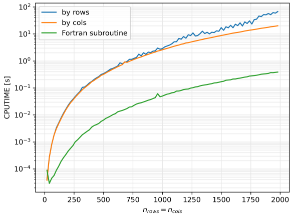
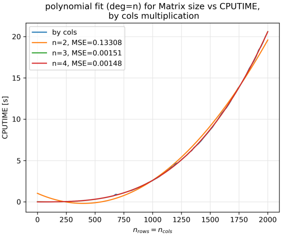
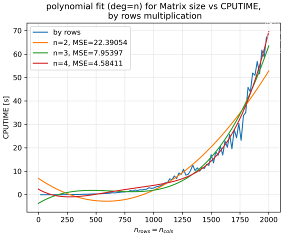


Figure: n vs CPUTIME



(a) polynomial fit, by columns multiplication



(b) polynomial fit, by rows multiplication

a_4	a_3	a_2	a_1	a_0	n=2	n=3	n=4
$-7.3 \cdot 10^{-14}$	$2.8 \cdot 10^{-9}$	$-9.8 \cdot 10^{-8}$	$3.9 \cdot 10^{-6}$	$1.3 \cdot 10^{-3}$	$1.33 \cdot 10^{-1}$	$1.51 \cdot 10^{-3}$	$1.48 \cdot 10^{-3}$

a_4	a_3	a_2	a_1	a_0	n=2	n=3	n=4
$2.5 \cdot 10^{-11}$	$-7.4 \cdot 10^{-8}$	$8.0 \cdot 10^{-5}$	$-2.9 \cdot 10^{-2}$	2.4	22.4	7.9	4.6

Table: Coefficients for $p(x)=a_4x^4 + a_3x^3 + a_2x^2 + a_1x + a_0$ fit for CPUTIME scaling on matrix size and MSE for deg=n polynomial fit, performing multiplication by cols (top) and by rows (bottom)

The MSE for the $n = 4$ polynomial fit is, as expected, slightly smaller than the $deg = 3$ one. Despite this, we can observe $\frac{a_4}{a_3} \sim 10^{-4}$, so the $deg = 4$ polynomial is probably overfitting: applying Occams's razor principle we can assume that the matrix multiplication operation scales like $\mathcal{O}(n^3)$

Code

We exploit the 'BLAS' and 'LAPACK' libraries, which contain the 'ZHEEV' subroutine used to diagonalize Hermitian matrices.

To compile: *gfortran -llapack -lblas debugging.f90 matrices.f90*

It is interesting to dive into the zheev subroutine:

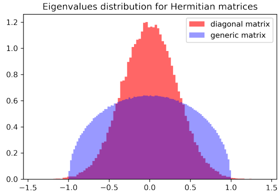
```
lwork=-1
!   zheev(jobz, uplo, N,  A, lda,   w, work, lwork, rowrk, info)
call zheev('N', 'U', N, matrix, N, eigv, dummy, lwork, rwork, info)
if (info == 0) then
   lwork = max((nb+1)*N, nint(real(dummy(1))))
endif

allocate (work(lwork))
call zheev('N', 'U', N, matrix, N, eigv, work, lwork, rwork, info)
```

In particular, when setting the lwork parameter to -1, we have that work(1) will store the optimal size of the work array

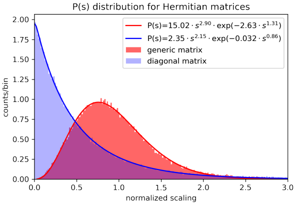
Results

We perform a fit of the normalized spacing distribution for both generic and diagonal Hermitian matrices with the function

$$P(s) = a \cdot s^{\alpha} \cdot \exp\left(-b \cdot s^{\beta}\right)$$



(a)



(b)

a	α	b	β
15.02	2.90	2.63	1.31

a	α	b	β
2.35	2.15	0.032	0.86

a	α	b	β
$\frac{32}{\pi^2}$	2	$\frac{4}{\pi}$	2

Table: Fit parameters for P(s) distribution for generic (top left) and diagonal (top right) Herimitian matrix and true values from Wigner sumrise (bottom)