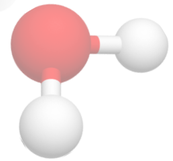
Université Paris Cité

Conception of a python program to calculate the Solvent-Accessible Surface Area (SASA) of a protein by using its atomic coordinates



2022/2023

M2 BI : Biologie-Informatique

Souad Youjil Abadi

souad.youjil-abadi@etu.u-paris.fr

INTRODUCTION

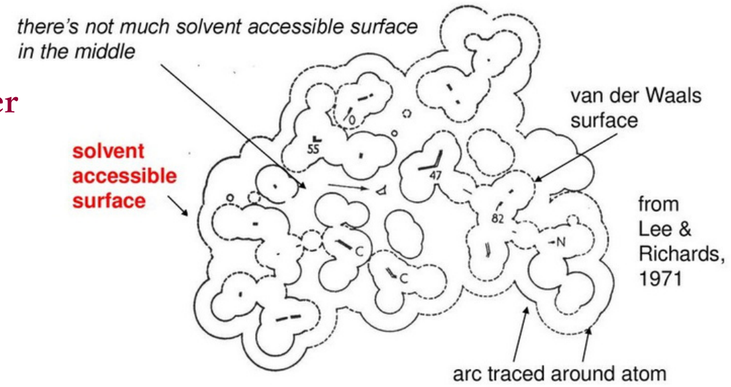
Lee & Richards
1971

Shrake & Rupley
1973

Firsts to :

- Described SASA (Solvent-Accessible Surface Area), Lee-Richards molecular surface
- Develop a computer program for calculating the SASA of protein atoms
- Describe the protein by a set of **solvated van der Waals spheres**
- Do not explicitly consider hydrogen atoms in H20 molecules (solvent)

The surface is approximated by the outline of a set of slices.



Cross-section (slice) of a protein

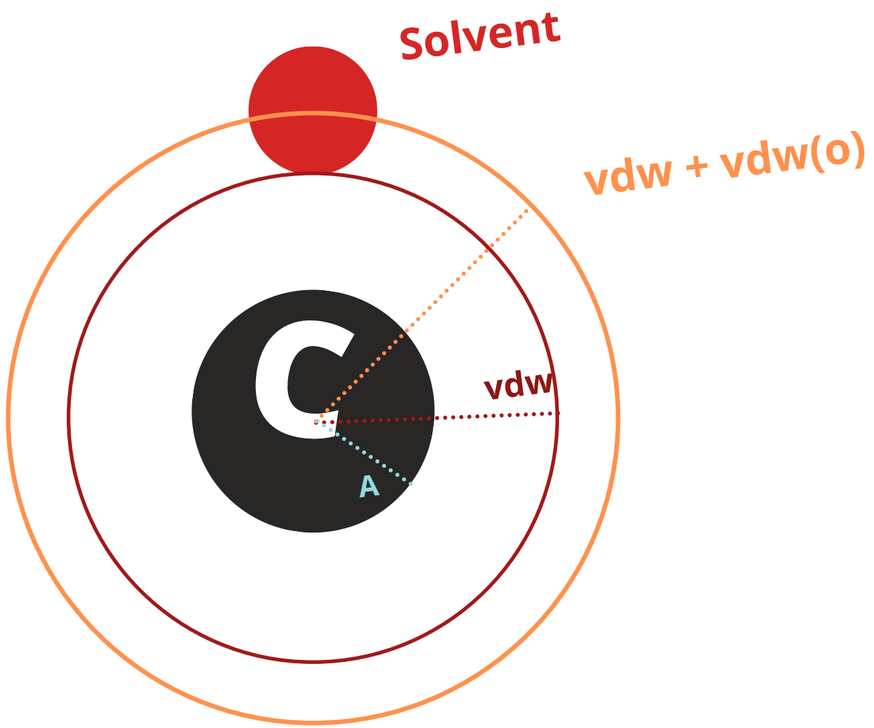
Based on Lee & Richards assumptions.

The surface of a sphere is represented by a set of 92 test points that are uniformly distributed.

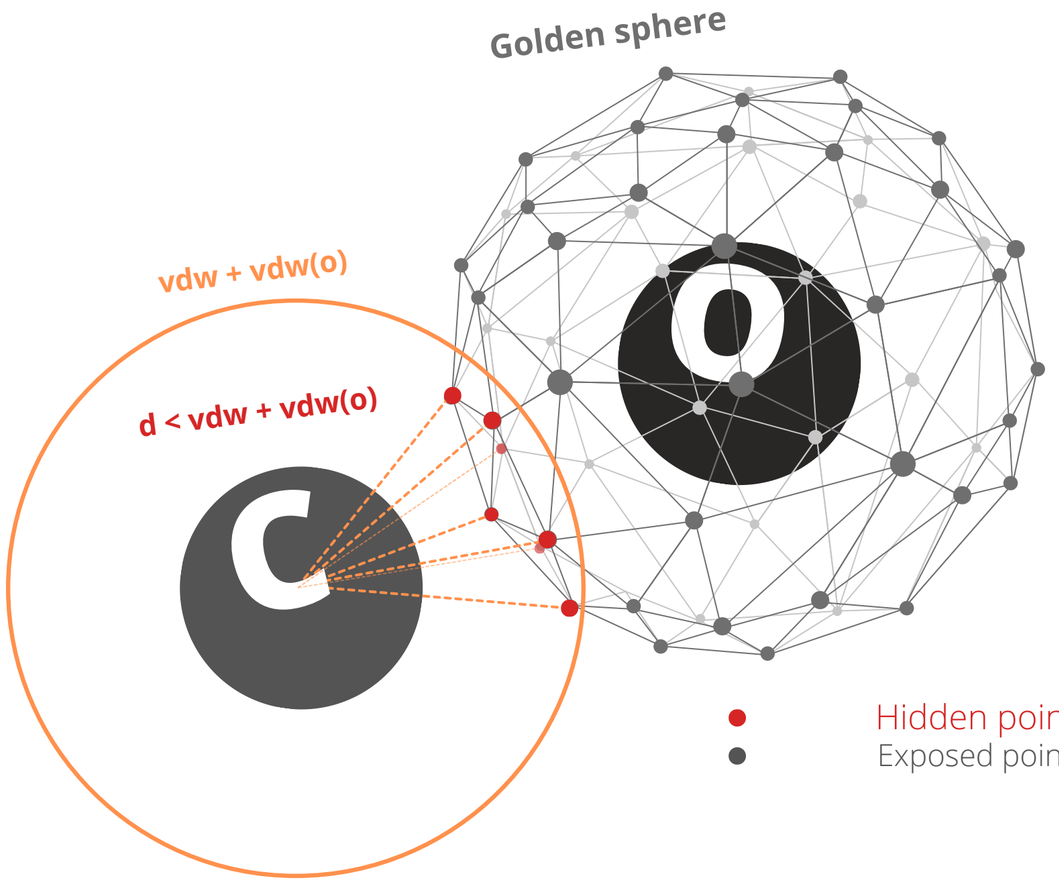
Why 92?

- If less: The **sphere level of detail decreases**
- If more: The **program execution time increases**
  ++ Increasing the number of points from 92 to 400 on a sphere, doesn't significantly change the results

Solvated van der Waals spheres (Shrake & Rupley 1973)



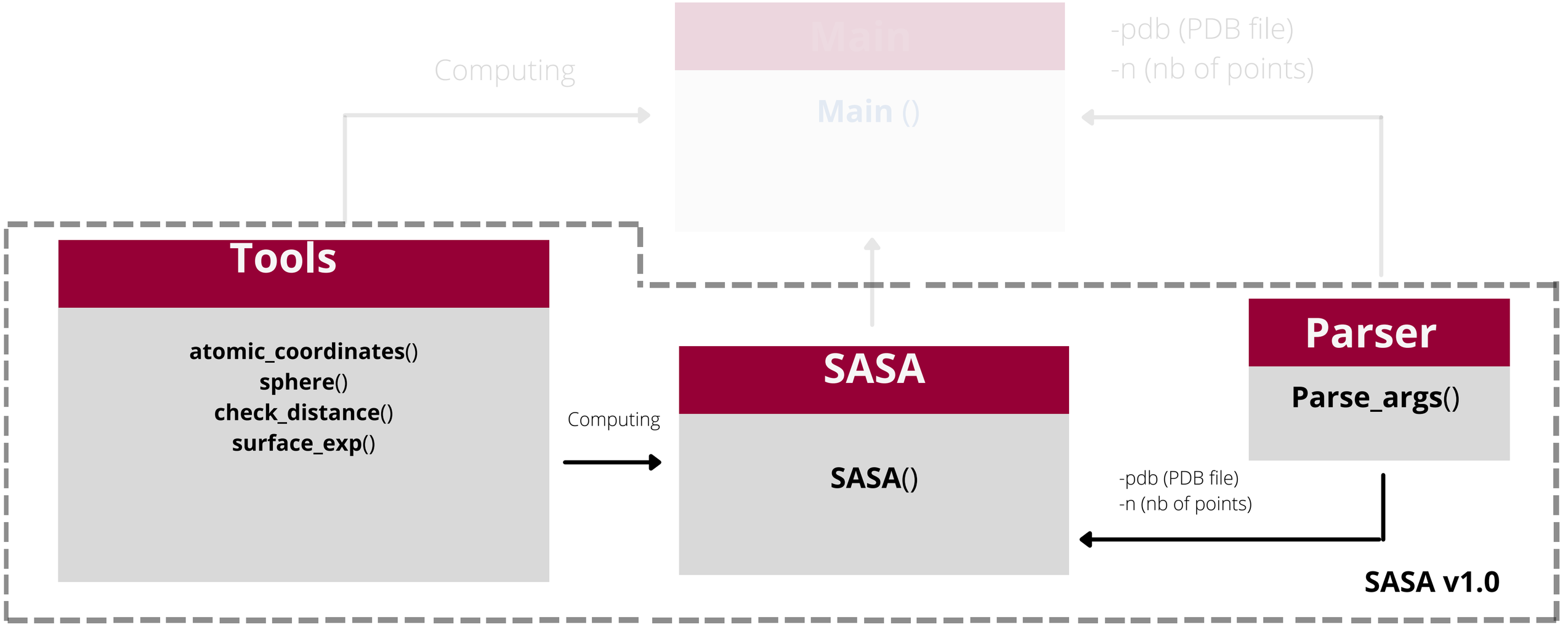
A : Atomic radius
vdw : Atom van der Waals radius
vdw(o) : Oxygen (solvent) van der Waals radius (1.4 Å)



Accessibility percentage = (Number of exposed points / total number of points) X 100

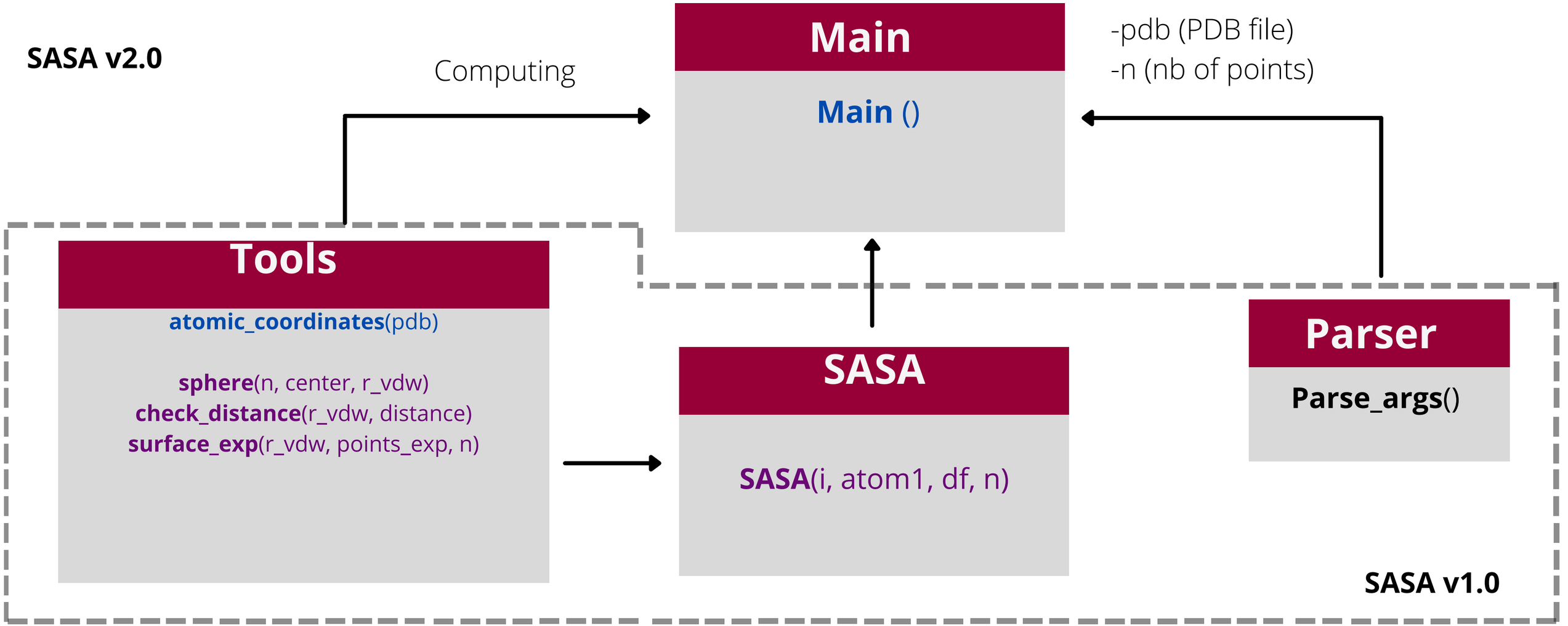
SASA PROGRAM

OBJECTIVE: Develop a program able to use the atomic coordinates of a protein -from a PDB file- to calculate the ASA, the relative ASA (rASA) and the accessibility percentage per residue.



SASA PROGRAM (after parallelization using Joblib)

The function SASA() is used by the main in a parallelized way (i.e. runned in parallel): The DataFrame is divided by n and SASA() is parallely runned on each part.



S A S A v2.0

Parses the **PDB** file
Extracts the atom data

atomic_coordinates(pdb)

SASA(i, atom1, df, n)

sphere(n, center, r_vdw)

Evenly distributes the number of points entered by the user on a sphere around the atom

Distance between an atom and the other protein atoms is then calculated and only the atoms located in the neighborhood (**d<10Å**) are selected

check_distance(r_vdw, distance)

Distance between the points of an atom and the nearest selected atoms is calculated and checked (**d < VdW radius of the atom + 1.4Å**)

Converts the number of the exposed points of an atom to Å² : **exposition ratio** (points exposed/total points) *** (4 * π * (sphere radius)²**).

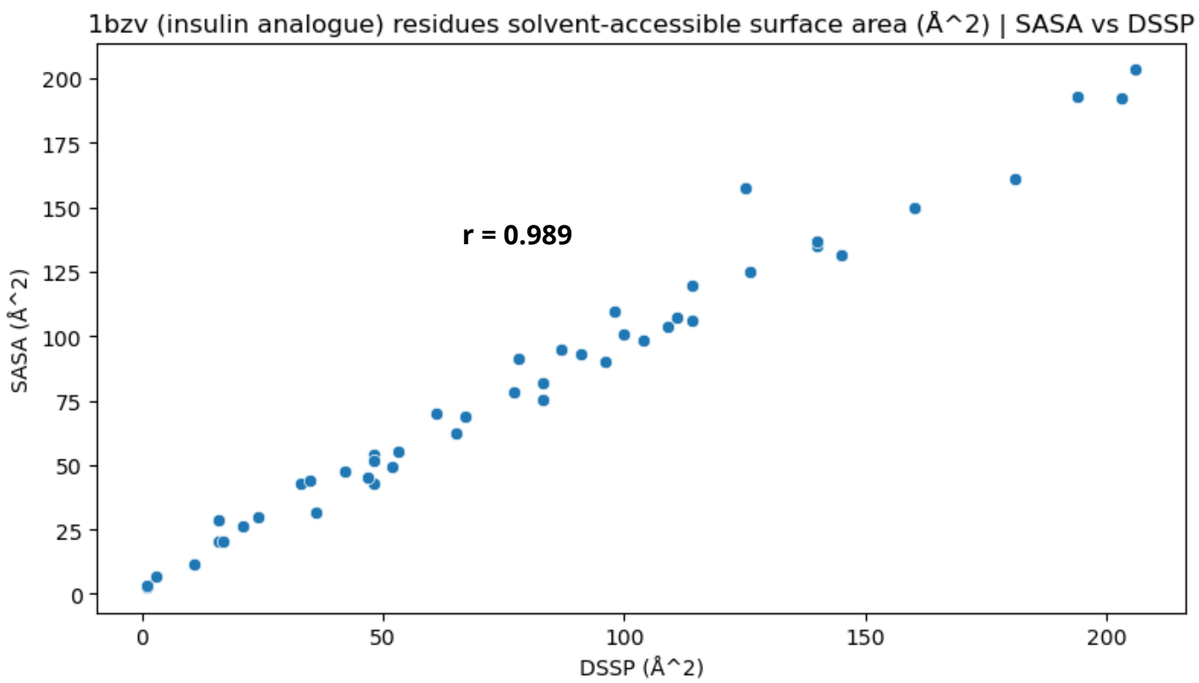
surface_exp(r_vdw, points_exp, n)

main()

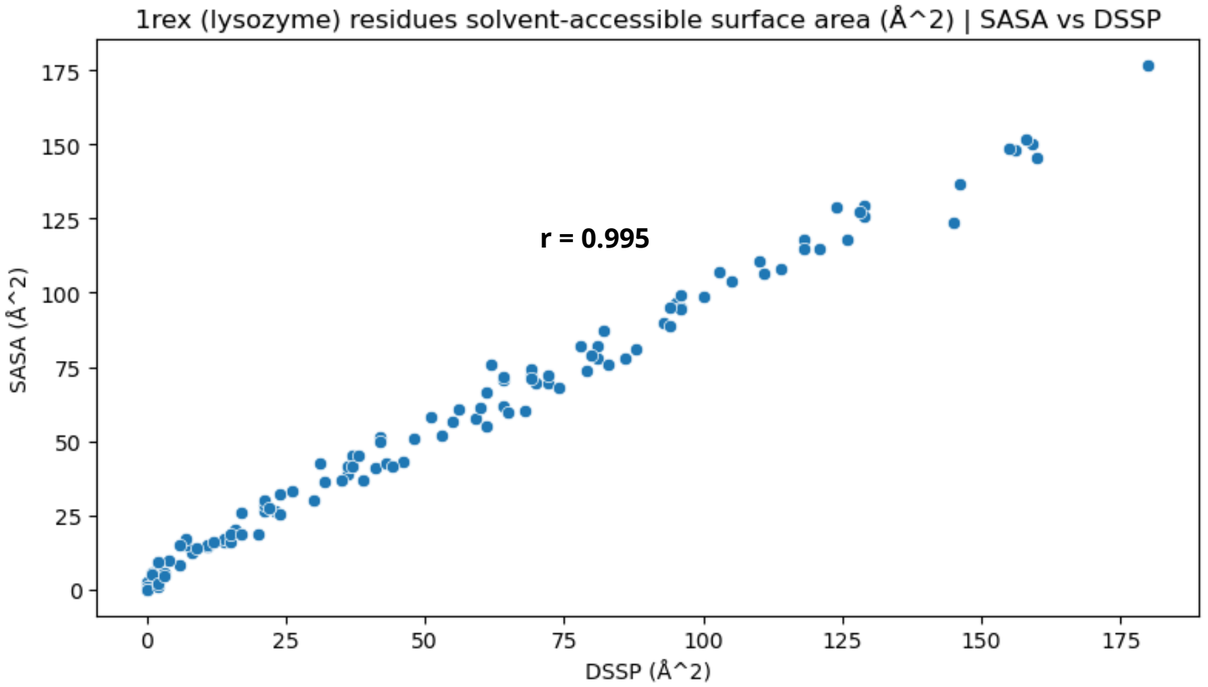
SASA() is used by the main in a parallelized way (i.e. runned in parallel):
The DataFrame is divided by n and SASA() is parallely runned on each part.

RESULTS

Measurements



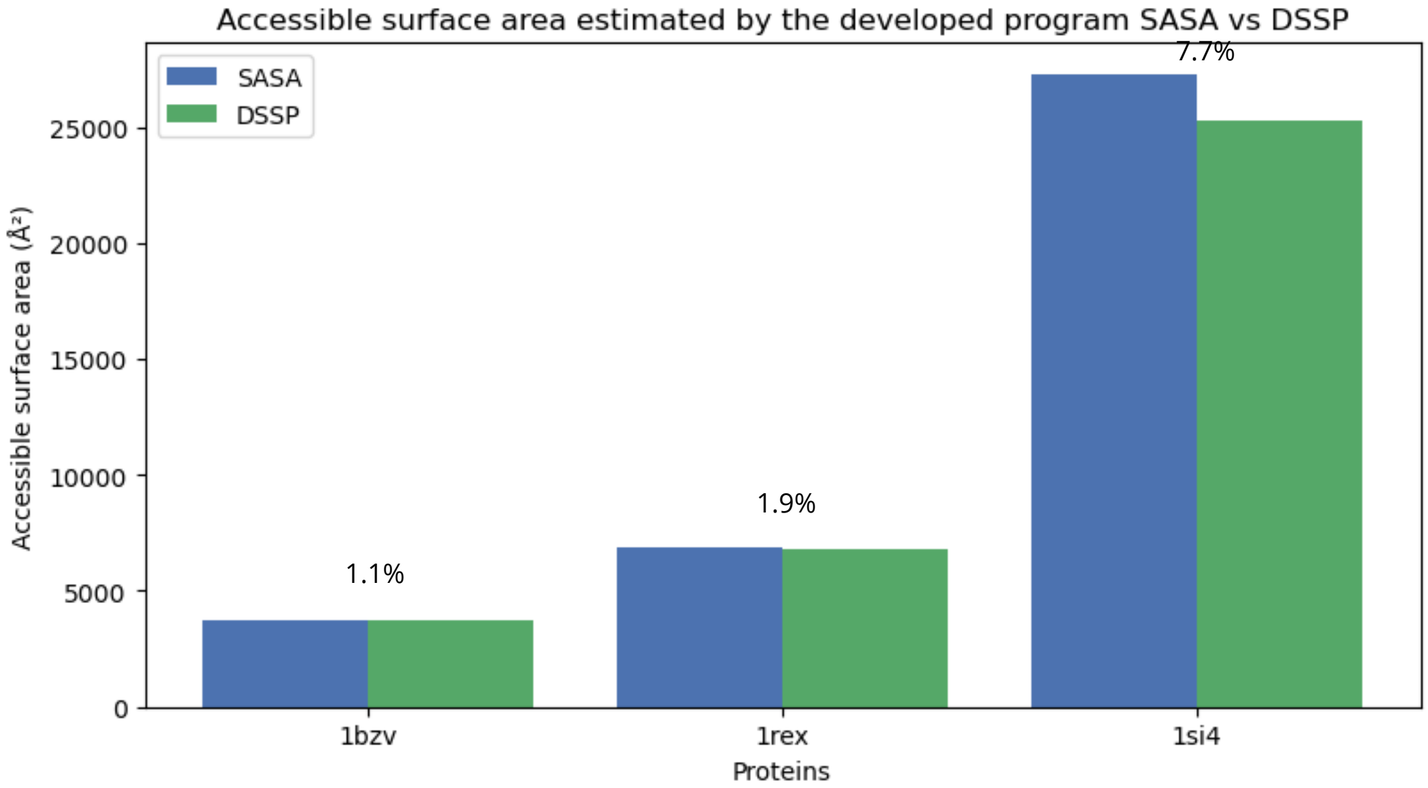
1bzv protein (46 residues)



1 rex protein (130 residues)

RESULTS

Measurements



- 1bzv : 366 atoms
- 1rex : 1135 atoms
- 1si4 : 4379 atoms

RESULTS

Time of execution

Timing	SASA v1.0	SASA v2.0
1bzv	8 sec	4 sec
1rex	85 sec	41,6 sec
1si4	18 min	7 min

Timing before and after parallelization

Machine used to execute the program:
15.1Go RAM, 4 CPUs
Lenovo Thinkpad W540 Corei7 SSD250

CONCLUSION & PERSPECTIVES

- The developed program provide **very similar results** to those obtained with DSSP for **small proteins**.

- For **large proteins**, the error may be decreased by increasing the number of points on the spheres.
  Indeed, DSSP uses a similar algorithm (named geodesic sphere integration) to the algorithm of Shrake and Rupley,
  however employs a **higher number of points** (360 vs 92).

- Conserning the **timing**:
  - DSSP uses a **recursive algorithm** while we use an **iterative algorithm** (**linear vs quadratic complexity**):
    For n = 10: 100ns (O(n)) vs 1000ns (O(n2)).
  - It is also important to note that the programming language used to write the program DSSP is C++,
    which is often **around 10x faster than Python**.

- The **time of execution was successfully increased** by employing multiple processors in parallel (simultaneously).

- More optimization can be done by **implementing OOP** (e.g. using each atom as an object) and **avoiding the use of the library Pandas**.