Ligand Docking Benchmark in Rosetta using Explicitly Placed Atomic Orbitals

By

Thomas Willcock

Thesis

Submitted to the Faculty of the

Graduate School of Vanderbilt University

in partial fulfillment of the requirements

for the degree of

MASTER OF SCIENCE

in

Chemistry

August, 2015

Nashville, Tennessee

Approved:

Jens Meiler, Ph.D.

Carmelo Rizzo, Ph.

Table of Contents

Page

LIST OF TABLES

LIST OF FIGURES

Figure Page

Introduction

Computational small-molecule docking plays a critical role in structure-based and ligand-based drug-design[1]. An example of successful drug design using small-molecule docking is the cancer drug Imatinb (Gleevec) for blood cancer[2]. Mainly, computational docking is used in early stages of drug discovery to scan millions of small molecules against target proteins as potential drug candidates[3]. Further, computational docking helps in optimizing the lead compounds for their ADME properties[4].

A variety of programs have been developed for small-molecule docking to biological molecules throughout the last few decades including Dock[5], AutoDock[6], Flexx[7], Gold[8], and RosettaLigand[9]. All these programs use a scoring potential that captures the chemical and physical properties of the small-molecule as well as the biomolecule, including the chemical interactions. Various algorithms that are used for docking in these programs include Monte Carlo minimization, genetic algorithms, Pose Clustering, and other methods[10,11,12,13]. Docking programs attempt to find the lowest energy structure of the ligand in the binding pocket.

Protein-ligand interactions are important for biological processes such as enzyme catalysis, protein activation by both natural and synthetic ligands, and protein inhibition through the use of synthetic drugs and drug-like molecules[14,15]. The ability to model these interactions at atomic level is crucial to understand the biochemistry underlying these processes. The ROSETTA molecular modelling suite[16] uses a knowledge-based potential to score the various energy terms and Monte Carlo algorithm with Metropolis criterion for minimization of ligand-protein docked complex.

The Rosetta energy function is a linear combination of energy terms that model various interactions between the atoms. The *Score12* function contains a van der Waals interactions (*fa_atr*), an inter side chain (*fa_rep*) and intra side chain repulsive term (*fa_intra_rep*), an implicit solvation model (*fa_sol*), a hydrogen bond term for side chain – side chain (*hbond_sc*), backbone – backbone (*hbond_sr_bb, hbond_lr_bb*) and backbone – side chain (*hbond_bb_sc*), a backbone –

dependent rotamer probability (*fa_dun*), the probability of an amino acid given *phi* and *psi* angles (*p_aa_pp*), the probability of two polar residues being within a certain distance of each other (*fa_pair),* and reference energies to resemble the quantity of residues seen in any given protein. The total energy of any given conformation is the sum of each of the terms multiplied by the weight that each term is given[17].

The newer *Talaris* score functions (*Talaris2013* and *Talaris2014*) are an update to the *Score12* score function. They have not been fully released yet but are used as the current internal standard. They contain a number of improvements to the *Score12* system, including the corrected internal coordinates of certain amino acid side chains and a better recovery of hydrogen bond acceptor - hydrogen bond donor distances and angles. The *Talaris* score functions also smoothen some of the statistical potentials and reweights them for use in protein folding and protein-protein docking[17,18,19].

Recent research shows that partial covalent interactions are important in ligand docking[20,21]. Partial Covalent Interactions (PCIs) are formed between orbitals of high and low electron density, and the lower free energy of the stable structures are a direct consequence of these interactions.

The existing method for capturing PCIs in Rosetta involves the Hydrogen Bonding potential which is a simple, orientation-dependent statistical energy term. This term only captures some of the PCIs, ignoring important classes such as π stacking interactions, T-stacked interactions, and cation- π interactions. It also incompletely describes the angular dependence of the interaction as the position of the orbital is not clearly defined, instead requiring the use of a torsion to attempt to capture the dependence implicictly[17,22].

The research described in this paper will use Rosetta as the framework from which to describe PCIs. Major part of the research deals with developing an extension to Rosetta, AssignOrbitals,that allows PCIs to be explicitly modelled and scored. Normally electron density distributions are calculated using resource-intensive quantum mechanical (QM) calculations. A method has been developed that can capture the location of electron pairs on atoms within a

molecule and the PCIs that they undergo without the need of computationally expensive QM calculations and this method is useful for a number of programs within Rosetta suite.

In the work described below, the existing Rosetta framework is updated in order to include the explicit placement of atomic orbitals using hybridization rules that are consistent with a quantum chemical picture of protein-ligand complexes. The top8000[23] dataset was used to create a statistical potential across hydrogen bonding interactions, π- π interactions, and cation- π interactions. This updates the existing score functions by replacing side chain – side chain (*hbond_sc*), backbone – backbone (*hbond_sr_bb, hbond_lr_bb*), and backbone – side chain (*hbond_bb_sc*) hydrogen bonding terms with terms for T-stacked cation-π, salt bridges, and hydrogen bonds *(orbitals_hpol)*, T-stacked and offset parallel π-π interactions *(orbitals_haro)*, and parallel π-π interactions and cation-π interactions (*orbitals_orbitals)*. Terms in italics are the scoring terms used by Rosetta for those interactions. The weights for the other energy terms have been adjusted accordingly in order to capture the chemical properties properly. In this paper, we discuss the addition of 3 different partial covalent interaction terms to score function and the results of a ligand docking benchmark data set that compares the performance of the newly developed Orbital Score function to capture the PCIs during ligand binding as compared to *Talaris2013*, *Talaris2014* and the *Score12* score function. Finally, we discuss about other possible areas for further improvement in score function.

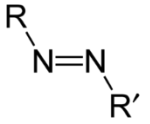
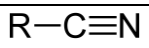
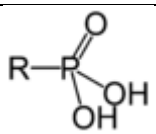
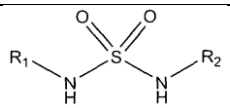
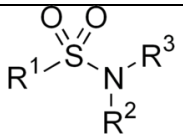
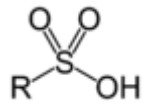
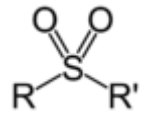
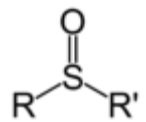
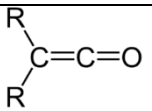
Results and discussion

Orbital Placement

Under consideration are two ways to place the orbitals on the atoms that will be used to create scoring terms. The first is the placement of a static orbital or set of orbitals on each atom that might be able to undergo PCIs. This would not change location or distance based on environment, but would rather be placed on the atom according to a specific set of rules. The second way is to place the orbitals while taking the environment into account, possibly even to allow the orbital

location to be able to move in space in order to better interact with another orbital or hydrogen in order to make the PCI.

Table 1: Important Functional Groups.

Functional groups common to Drug-like molecules found in the BCL	Image
Di-Nitrogen	
Nitrile	
Phosphoxide (Phosphonic Acid)	
Sulfamide	
Sulfonamide	
Sulfonate (Sulfonic Acid)	
Sulfone	
Sulfoxide	
Ketene	

In the first case, where environment does not play any role in defining PCIs, the orbital locations can be placed based only on the properties that are explicitly known about the atom in question. For example, a hydrogen bond from an alcohol group will not be different if it is formed in a ligand binding site, inside a protein or in a protein-protein interface. In order to understand the changes that orbitals undergo upon binding, a series of QM calculations were performed on ligands from the benchmark dataset in their free and bound form. Based on these calculations, rules were developed to place orbitals on the atom for PCI term calculation during ligand binding. More information on how the calculations were

carried out is included in the methods section.

The benchmark set that is being used was originally used as a ligand set redesign benchmark[24]. It was chosen because of the ligand diversity, the quality of the crystal structures, and the size of the protein-ligand complexes.

A search was done in the Biochemical Library (BCL) for drug-like targets to collect a set of functional groups that are important in ligand-docking. Table 1 shows a partial list of functional groups found in drug-like molecules in the BCL. Other well-known functional groups ,such as, general double or triple bonds, alcohols, esters, ethers, carboxylic acid were also tested (not shown).. Table 1 shows more complicated groups that require special attention. For example, the oxygen-sulphur bond needs to be tested to show what the orbitals look like as it is not a usual double bond, but has some single bond characteristics as well.

The quantum orbitals of each of the functional groups were calculated using a single point energy calculation. Similar orbital calculations were performed for each of the ligands in the benchmark set. The orbitals on the ligands and on the functional groups in question were compared. Despite the differences in environment, the orbitals were unchanged. These orbitals were used to identify single point locations that could represent the quantum orbitals. These points were always within the quantum orbitals.

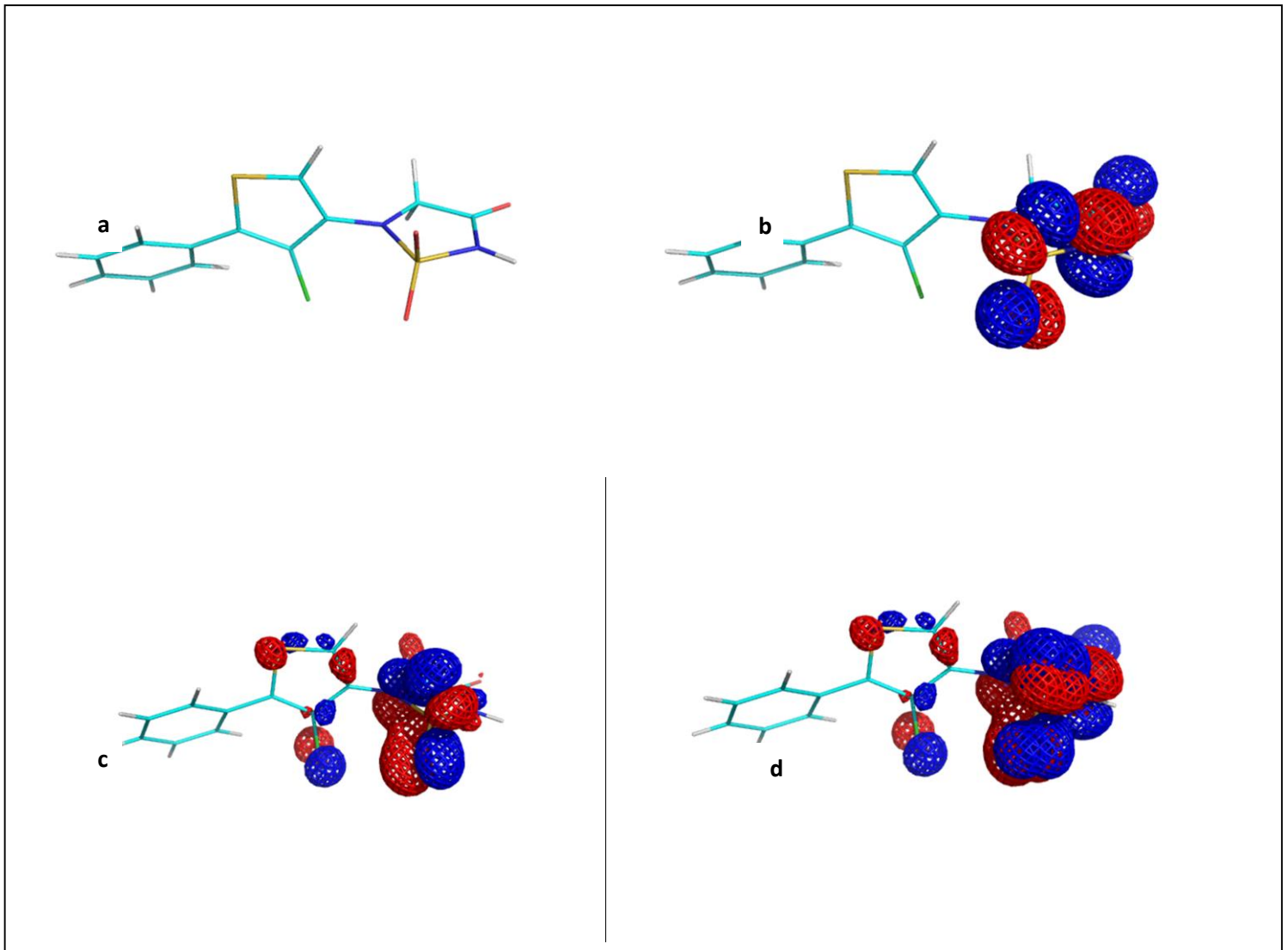


Figure 1: **A quantum view of Sulfones.** From Top left to bottom right are the quantum mechanical results for one of the gas-phase benchmark sets, in this case, sulfone. They form two different sets of perpendicular orbitals that form a cross. The two crosses are parallel to each other across the O=S=O bond.

For reference, Figures 1 and 2 are provided over the next two pages. The red and blue lobes are the positive and negative lobes on the orbitals that were calculated for the molecules in question. Looking at the oxygen atoms in the sulfone, four different lobes can be seen. These four lobes form a cross around the oxygen with no electron density extending from the end. There are then two crosses, one on each oxygen atom, made up of four lobes each. Further, each cross is

parallel to the cross on the other oxygen atom and in line with the Oxygen-Sulphur-Oxygen bonding set.

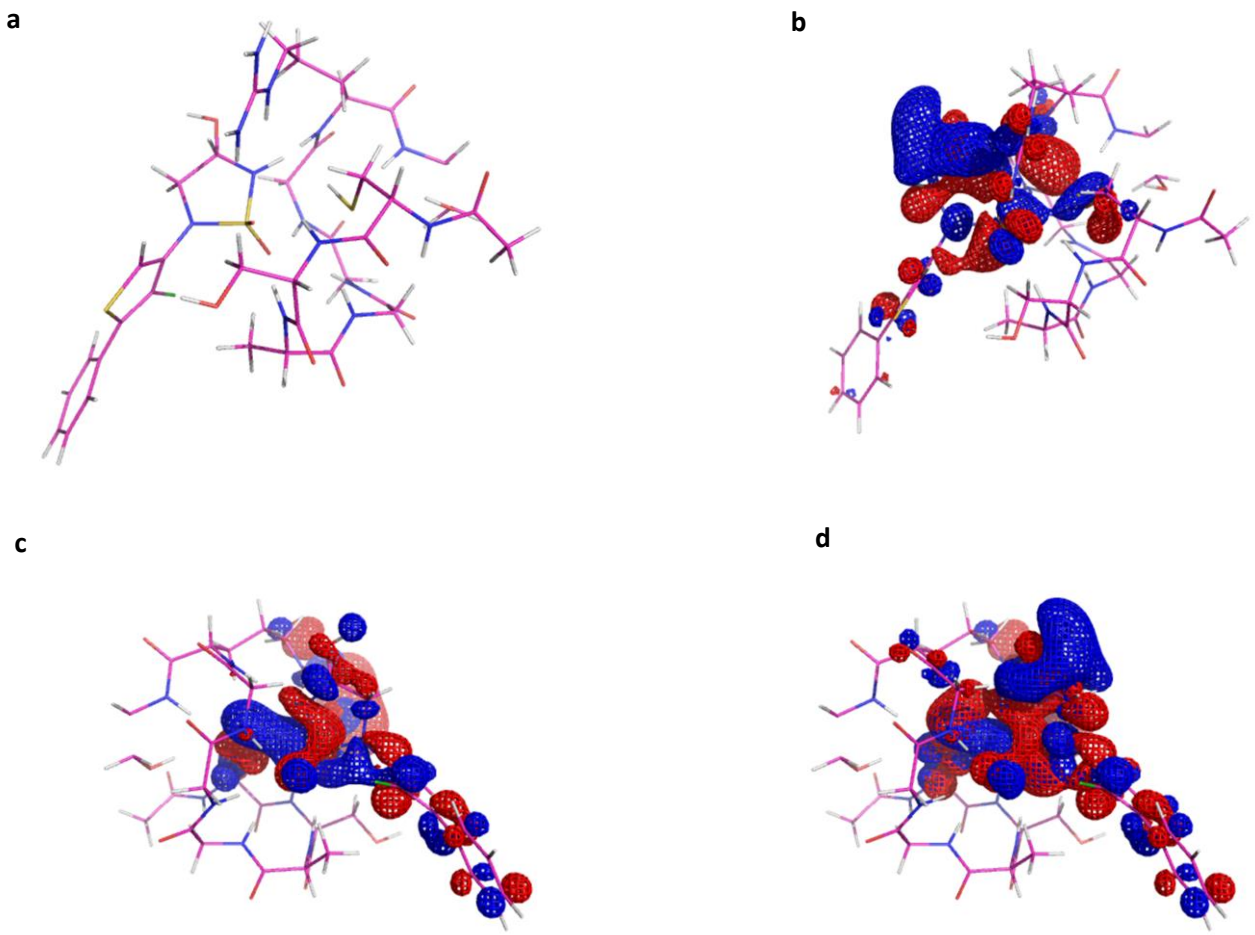


Figure 2: **A quantum view of Sulfones with interacting partner.** From top left to bottom right. The results of the quantum mechanical calculations looking at the orbitals on the sulphur when the interacting residues are there. Each sulphur has two sets of lobes that are perpendicular to each other.

This particular set up was the same for all sulfones. A variety were tested in the gas phase to make sure that it was set up correctly and to see if the environment had an effect on the shape and orientation of the orbitals. The environment had no effect on the number of lobes, or their orientation, but it did have a small effect on the size of the lobes. The change in size of the lobes was ultimately neglected because the variation was always between one and two Van Der Waals radii.

Orbital size is dependent on electron density, and was attributed to the existence of an interaction between the protein and the ligand.

This pattern was seen through each of the ligand-protein complexes, where the sets of orbitals that were important for PCIs was consistent in shape and orientation between when it was in the presence of the interacting residues and when it wasn't. The only difference was the size of the orbitals in this case, and that was not too big of a difference.

In light of this result, the orbital placement was set up without taking into account the environment around the orbital, but only the properties of the atom in question. These properties should also be consistent with an orbital view of the atom, and so the atoms were typed using Gasteiger atom types instead of normal Rosetta atom types[25]. A second reason to use the Gasteiger atom types is that the Rosetta atom types were originally created with proteins in mind and so typing is based atoms that appear only in proteins. Many atom types that appear in ligands do not appear in proteins, for example, the sulphur in the sulfone group does not appear and so have not been typed in Rosetta.

Assign Orbital Code

Based on the above calculations, a program was written, AssignOrbitals, which places a single point where each orbital lobe would exist. Each atom that contains π orbitals or lone pairs will therefore have orbital points that could undergo a PCI. The code first parses the protein or protein-ligand complex, and it assigns the Gasteiger atom types to each atom in the protein and ligand. This means that the important information of hybridization and number of bonds and number of lone pairs is captured for each atom.

AssignOrbitals cycles through each of the atoms to check how many bonds it has, consider the hybridization of the atom looking at sp, sp2 and sp3 hybridized atoms, and finally determines the number of lone pairs present. With this information, orbitals are placed where the middle of the orbital lobe will be in Quantum Mechanical calculations. The orbitals are placed at a fixed distance away from the atom based on the Bohr Radius of that atom[26]. There are a few cases that require slightly different orbital placement from the normal. For example, the oxygen that is double bonded to a sulphur in the sulfone group is an sp2 hybrid oxygen that has a single bond. But in order to be consistent with the QM view of the system (seen in Figure 1 and 2), the program will place the orbitals differently on this atom than on other sp2 hybridized oxygen with one bond.

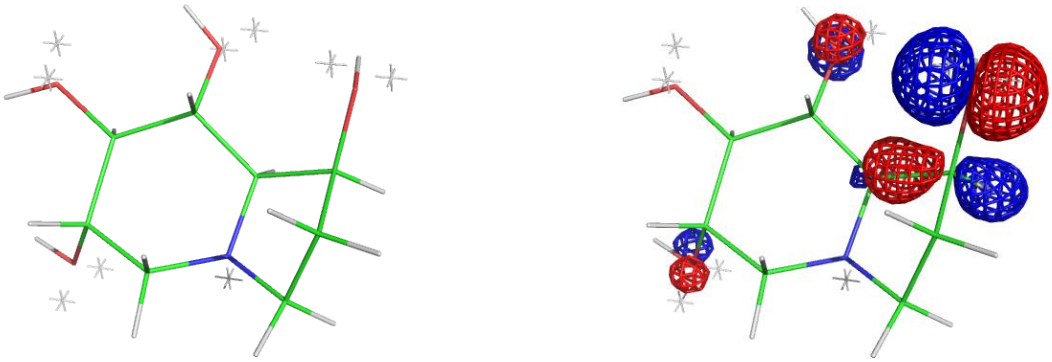


Figure 3: **AssignOrbital Placement of Atomic Orbitals.** On the left is a simple carbohydrate showing as white crosses the orbitals that are placed on the ligand. Each sp3 oxygen gets two orbitals that correspond to the two lone pairs that are present on the oxygen. On the right is the same carbohydrate with the orbitals from the QM calculations shown on the oxygen in the foreground. As expected, the orbitals are inside the lobes that are calculated. The placed orbitals are close to the edge, about 4/5 of the full distance from the orbital, and close to the center. The angles are 107° as is consistent with VSEPR theory.

In general, however, the orbitals are placed in a way that is consistent with an organic chemist's view of hybridization and orbitals. Figure 3 shows an example where orbitals are placed on sp3 oxygens in an alcohol functional group.

Once the orbitals are placed they are used to create statistics on how PCIs should look. These statistics are used in the score function. The earlier mention of hydrogen- π, π - π, and cation-π interactions refers to the statistics mentioned here.

	-0.975	-0.925	-0.875	-0.825	-0.775	-0.725	-0.675	-0.625	-0.575	-0.525	-0.475	-0.425	-0.375	-0.325	-0.275	-0.225	-0.175	-0.125	-0.075	-0.025	0.025	0.075	0.125	0.175	0.225	0.275	0.325	0.375
0.05	5.136016	5.136016	5.136016	5.136016	5.136016	5.136016	5.136016	5.136016	5.136016	5.136016	5.136016	5.136016	5.136016	5.136016	5.136016	5.136016	5.136016	5.136016	5.136016	5.136016	5.136016	5.136016	5.136016	5.136016	5.136016	5.136016	5.136016	5.136016
0.15	5.136016	5.136016	5.136016	5.136016	5.136016	5.136016	5.136016	5.136016	5.136016	5.136016	5.136016	5.136016	5.136016	5.136016	5.136016	5.136016	5.136016	5.136016	5.136016	5.136016	5.136016	5.136016	5.136016	5.136016	5.136016	5.136016	5.136016	5.136016
0.25	5.136016	5.136016	5.136016	5.136016	5.136016	5.136016	5.136016	5.136016	5.136016	5.136016	5.136016	5.136016	5.136016	5.136016	5.136016	5.136016	5.136016	5.136016	5.136016	5.136016	5.136016	5.136016	5.136016	5.136016	5.136016	5.136016	5.136016	5.136016
0.35	5.136016	5.136016	5.136016	5.136016	5.136016	5.136016	5.136016	5.136016	5.136016	5.136016	5.136016	5.136016	5.136016	5.136016	5.136016	5.136016	5.136016	5.136016	5.136016	5.136016	5.136016	5.136016	5.136016	5.136016	5.136016	5.136016	5.136016	5.136016
0.45	1.833798	5.136016	5.136016	5.136016	5.136016	5.136016	5.136016	5.136016	5.136016	5.136016	5.136016	5.136016	5.136016	5.136016	5.136016	5.136016	5.136016	5.136016	5.136016	5.136016	5.136016	5.136016	5.136016	5.136016	5.136016	5.136016	5.136016	5.136016
0.55	5.136016	5.136016	5.136016	5.136016	5.136016	5.136016	5.136016	5.136016	5.136016	5.136016	5.136016	5.136016	5.136016	5.136016	5.136016	5.136016	5.136016	5.136016	5.136016	5.136016	5.136016	5.136016	5.136016	5.136016	5.136016	5.136016	5.136016	5.136016
0.65	5.136016	5.136016	5.136016	5.136016	5.136016	5.136016	5.136016	5.136016	5.136016	5.136016	5.136016	5.136016	5.136016	5.136016	5.136016	5.136016	5.136016	5.136016	5.136016	5.136016	2.530044	5.136016	5.136016	5.136016	5.136016	5.136016	5.136016	5.136016
0.75	5.136016	5.136016	5.136016	5.136016	5.136016	5.136016	5.136016	5.136016	5.136016	5.136016	5.136016	5.136016	5.136016	5.136016	5.136016	5.136016	5.136016	5.136016	5.136016	5.136016	5.136016	5.136016	5.136016	5.136016	5.136016	5.136016	5.136016	5.136016
0.85	5.136016	5.136016	5.136016	5.136016	5.136016	5.136016	5.136016	5.136016	5.136016	5.136016	5.136016	5.136016	5.136016	5.136016	5.136016	5.136016	5.136016	5.136016	5.136016	5.136016	5.136016	5.136016	5.136016	5.136016	5.136016	5.136016	5.136016	5.136016
0.95	5.136016	5.136016	5.136016	5.136016	5.136016	5.136016	5.136016	5.136016	5.136016	5.136016	5.136016	5.136016	5.136016	5.136016	5.136016	5.136016	5.136016	5.136016	5.136016	5.136016	5.136016	5.136016	5.136016	5.136016	5.136016	5.136016	5.136016	5.136016
1.05	3.376908	5.136016	5.136016	5.136016	5.136016	3.376908	5.136016	3.376908	5.136016	5.136016	5.136016	5.136016	5.136016	5.136016	5.136016	5.136016	3.376908	5.136016	5.136016	5.136016	5.136016	5.136016	5.136016	5.136016	5.136016	5.136016	5.136016	5.136016
1.15	3.525067	5.136016	5.136016	3.525067	5.136016	5.136016	5.136016	5.136016	5.136016	5.136016	5.136016	5.136016	5.136016	5.136016	5.136016	5.136016	5.136016	5.136016	5.136016	5.136016	5.136016	5.136016	5.136016	5.136016	5.136016	5.136016	5.136016	5.136016
1.25	3.656231	3.083946	3.656231	5.136016	5.136016	5.136016	5.136016	5.136016	5.136016	5.136016	5.136016	5.136016	5.136016	5.136016	5.136016	5.136016	5.136016	5.136016	5.136016	5.136016	5.136016	5.136016	5.136016	5.136016	5.136016	5.136016	5.136016	5.136016
1.35	2.86139	3.772966	3.772966	3.772966	5.136016	5.136016	5.136016	5.136016	5.136016	5.136016	5.136016	5.136016	5.136016	5.136016	5.136016	5.136016	5.136016	5.136016	5.136016	5.136016	5.136016	5.136016	5.136016	5.136016	5.136016	5.136016	5.136016	5.136016
1.45	1.616428	2.525649	2.210403	3.337344	3.877325	2.988632	5.136016	3.877325	5.136016	5.136016	5.136016	5.136016	5.136016	5.136016	5.136016	5.136016	5.136016	5.136016	5.136016	5.136016	5.136016	5.136016	5.136016	5.136016	5.136016	5.136016	5.136016	5.136016
1.55	1.606996	2.210262	3.105488	2.648598	3.970981	3.970981	3.970981	3.970981	5.136016	5.136016	5.136016	3.970981	5.136016	5.136016	5.136016	5.136016	5.136016	3.970981	5.136016	5.136016	5.136016	5.136016	5.136016	5.136016	5.136016	5.136016	5.136016	5.136016
1.65	-0.13044	0.845154	1.230645	2.117182	2.117182	2.453134	2.453134	5.136016	5.136016	4.055316	4.055316	4.055316	3.548112	5.136016	3.548112	5.136016	4.055316	5.136016	5.136016	5.136016	5.136016	5.136016	5.136016	5.136016	5.136016	5.136016	5.136016	5.136016
1.75	-1.40537	-0.49124	0.330825	1.135085	1.563326	2.138449	2.868741	3.312784	3.312784	2.703836	3.640593	3.640593	5.136016	3.312784	5.136016	3.640593	4.13149	5.136016	5.136016	5.136016	5.136016	5.136016	5.136016	5.136016	5.136016	5.136016	5.136016	5.136016
1.85	-2.45017	-1.64105	-0.7024	-0.21137	0.308614	0.661863	1.309361	1.948481	2.541612	2.541612	4.200481	3.405117	4.200481	4.200481	4.200481	4.200481	3.405117	4.200481	5.136016	5.136016	5.136016	5.136016	4.200481	5.136016	5.136016	4.200481	5.136016	5.136016
1.95	-3.04251	-2.2983	-1.38782	-0.85453	-0.49793	0.152702	0.681401	1.200716	1.285952	2.185299	2.114934	2.530044	3.490929	3.252699	3.804273	3.490929	3.804273	5.136016	5.136016	5.136016	5.136016	5.136016	3.804273	4.263126	5.136016	5.136016	5.136016	5.136016
2.05	-3.19975	-2.51106	-1.7576	-1.3122	-0.83514	-0.58409	-0.12509	0.562876	0.956735	1.19548	1.7346	1.7346	3.147351	2.850772	2.622329	3.336848	3.147351	3.570851	4.320141	4.320141	3.876899	5.136016	5.136016	5.136016	5.136016	5.136016	5.136016	5.136016
2.15	-3.00974	-2.4302	-1.70133	-1.4008	-1.04308	-0.75925	-0.60609	-0.34248	0.058516	0.644176	0.766993	1.264623	1.91403	2.173579	2.613038	2.299753	3.071765	3.944177	3.415705	3.229021	3.229021	4.372147	4.372147	5.136016	3.944177	4.372147	5.136016	5.136016
2.25	-2.63999	-2.01065	-1.51236	-1.2087	-1.01898	-0.88353	-0.69044	-0.45873	-0.33017	-0.11359	0.10687	0.568868	0.814126	1.353567	2.259596	1.833798	2.385111	3.016345	2.897936	3.016345	3.489734	5.136016	5.136016	4.419683	4.006607	4.006607	5.136016	5.136016
2.35	-2.36457	-1.73166	-1.25426	-1.09014	-0.90517	-0.87974	-0.67805	-0.566	-0.47198	-0.46492	-0.2995	0.029401	0.133444	0.551755	0.832789	1.391996	1.843437	1.741799	2.34146	2.284438	2.688741	2.975261	3.559341	4.463219	3.378408	5.136016	5.136016	5.136016
2.45	-2.19951	-1.43671	-1.16135	-0.93826	-0.70343	-0.50198	-0.57471	-0.60919	-0.53598	-0.50198	-0.44374	-0.29685	-0.06317	0.123907	0.391932	0.614821	1.115249	1.215111	1.673291	1.619417	2.119245	2.543637	2.944732	3.446875	3.446875	3.841591	5.136016	5.136016
2.55	-2.02609	-1.20518	-0.83792	-0.67377	-0.53226	-0.45928	-0.38056	-0.45625	-0.48026	-0.42227	-0.33017	-0.26246	-0.2951	-0.18983	-0.07661	0.2515	0.471218	0.80546	1.088765	1.276149	1.837136	1.931764	2.494024	2.836858	2.922184	4.168978	5.136016	5.136016
2.65	-1.93265	-1.08435	-0.76119	-0.42416	-0.39469	-0.26726	-0.22917	-0.12118	-0.27403	-0.22917	-0.20415	-0.33618	-0.18957	-0.10932	-0.22563	-0.11329	0.168695	0.2451	0.618955	1.043875	1.269148	1.720457	1.852837	2.687943	2.267903	4.215959	5.136016	5.136016
2.75	-1.83589	-0.87912	-0.51835	-0.40682	-0.3119	-0.22692	-0.15544	0.086912	0.017448	-0.08605	-0.01555	-0.10479	-0.06696	-0.1448	-0.07464	-0.09359	0.091418	0.132913	0.191061	0.597957	0.833493	1.195221	1.390077	1.98215	2.255704	3.360084	5.136016	5.136016
2.85	-1.64679	-0.69956	-0.34114	-0.10877	-0.02629	0.080092	0.113917	0.035375	0.101098	0.271994	0.059518	0.232489	0.043358	0.063599	-0.09839	0.088441	0.144483	0.15339	0.166902	0.472069	0.429902	0.903928	1.12581	1.186781	1.680712	4.300928	5.136016	5.136016
2.95	-1.26542	-0.42168	-0.13215	0.178129	0.152773	0.217412	0.320479	0.440954	0.345462	0.365908	0.457794	0.355633	0.429882	0.413501	0.463472	0.345462	0.386781	0.360758	0.371086	0.397384	0.603824	0.528146	0.962442	1.2419	1.727083	5.136016	5.136016	5.136016
3.05	-0.68787	0.05048	0.206539	0.411562	0.292519	0.468794	0.442373	0.437171	0.61871	0.663231	0.69629	0.682935	0.676324	0.656748	0.650307	0.682935	0.631229	0.682935	0.709826	0.7804	0.864255	0.848476	1.056558	1.037465	1.677622	5.136016	5.136016	5.136016
3.15	-0.19239	0.421392	0.582049	0.645886	0.495915	0.701328	0.766766	0.733512	0.726992	0.993513	1.019294	0.976688	0.976688	0.993513	1.028038	0.943864	1.028038	0.836787	0.993513	1.091481	0.976688	1.045757	1.129627	1.321968	2.004891	5.136016	5.136016	5.136016
3.25	0.34501	1.005395	0.981488	0.898417	1.089483	0.898417	0.90569	0.973644	1.098294	1.152856	1.250981	0.973644	1.005395	1.261345	1.261345	1.134336	1.348363	1.143553	1.394887	1.107183	1.406864	1.549051	1.636072	1.383051	2.088118	5.136016	5.136016	5.136016
3.35	0.779573	1.37408	1.331117	1.231151	1.149002	1.123052	1.260106	1.279886	0.987514	1.407559	1.396275	1.407559	1.396275	1.442198	1.490332	1.607934	1.502736	1.580579	1.594163	1.502736	1.490332	1.694797	1.694797	1.694797	2.703294	5.136016	5.136016	5.136016
3.45	1.15552	1.678922	1.388597	1.637677	1.651237	1.388597	1.269935	1.388597	1.464932	1.378153	1.830163	1.782342	1.766897	1.598066	2.089695	1.487857	1.813967	1.863363	1.813967	2.028132	2.008425	1.970139	2.155298	1.951533	3.097492	5.136016	5.136016	5.136016
3.55	1.444358	1.837435	1.734217	1.935268	1.791888	1.837435	1.640663	1.627985	1.734217	1.734217	1.720304	1.777156	1.970128	1.627985	1.791888	1.885156	1.988025	2.279459	1.952546	2.043721	2.043721	2.187181	2.14405	1.918283	4.003725	5.136016	5.136016	5.136016
3.65	1.609087	2.356124	2.261967	1.906508	1.875504	2.155514	1.890886	2.059312	1.906508	1.922378	1.81624	1.720294	2.175918	2.041134	2.115916	2.09669	1.988503	2.218017	1.971557	2.218017	2.607342	2.356124	2.40675	2.307938	5.136016	5.136016	5.136016	5.136016
3.75	1.78566	2.227101	2.431647	2.148092	2.38244	2.040177	2.074871	2.092679	1.839982	1.912316	2.4834	2.110811	2.227101	2.227101	2.510317	2.247869	2.269077	2.4834	2.656978	2.510317	2.828908	2.510317	2.537979	2.756552	5.136016	5.136016	5.136016	5.136016
3.85	2.179243	2.340243	2.256507	2.318643	2.384897	2.198007	2.179243	2.142741	2.107524	2.198007	2.179243	2.340243	2.318643	2.455854	2.318643	2.5866	2.455854	2.953745	2.340243	2.559047	2.532234	2.839538	2.876177	3.082697	5.136016	5.136016	5.136016	5.136016
3.95	2.122332	2.432827	2.579632	2.661997	2.455846	2.345709	2.227796	2.410326	2.285016	2.246508	2.366788	2.366788	2.606341	2.633782	2.850382	2.455846	2.691032	2.691032	2.783565	2.885542	2.816416	2.816416	2.816416	3.385834	5.136016	5.136016	5.136016	5.136016

Figure 4: Donor-Hydrogen-Orbital Angles for π- π interactions represented as a heatmap.

	-0.975	-0.925	-0.875	-0.825	-0.775	-0.725	-0.675	-0.625	-0.575	-0.525	-0.475	-0.425	-0.375	-0.325	-0.275	-0.225	-0.175	-0.125	-0.075	-0.025	0.025	0.075	0.125	0.175	0.225	0.275	0.325	0.375	0.425
0.05	5.141501	5.141501	5.141501	5.141501	5.141501	5.141501	5.141501	5.141501	5.141501	5.141501	5.141501	5.141501	5.141501	5.141501	5.141501	5.141501	5.141501	5.141501	5.141501	5.141501	5.141501	5.141501	5.141501	5.141501	5.141501	5.141501	5.141501	5.141501	5.141501
0.15	5.141501	5.141501	5.141501	5.141501	5.141501	5.141501	5.141501	5.141501	5.141501	5.141501	5.141501	5.141501	5.141501	5.141501	5.141501	5.141501	5.141501	5.141501	5.141501	5.141501	5.141501	5.141501	5.141501	5.141501	5.141501	5.141501	5.141501	5.141501	5.141501
0.25	5.141501	5.141501	5.141501	5.141501	5.141501	5.141501	5.141501	5.141501	5.141501	5.141501	5.141501	5.141501	5.141501	5.141501	5.141501	5.141501	5.141501	5.141501	5.141501	5.141501	5.141501	5.141501	5.141501	5.141501	5.141501	5.141501	5.141501	5.141501	5.141501
0.35	5.141501	5.141501	5.141501	5.141501	5.141501	5.141501	5.141501	5.141501	5.141501	5.141501	5.141501	5.141501	5.141501	5.141501	5.141501	5.141501	5.141501	5.141501	5.141501	5.141501	5.141501	5.141501	5.141501	5.141501	5.141501	5.141501	5.141501	5.141501	5.141501
0.45	1.839284	5.141501	5.141501	5.141501	5.141501	5.141501	5.141501	5.141501	5.141501	5.141501	5.141501	5.141501	5.141501	5.141501	5.141501	5.141501	5.141501	5.141501	5.141501	5.141501	5.141501	5.141501	5.141501	5.141501	5.141501	5.141501	5.141501	5.141501	5.141501
0.55	5.141501	5.141501	5.141501	5.141501	5.141501	5.141501	5.141501	5.141501	5.141501	5.141501	5.141501	5.141501	5.141501	5.141501	5.141501	5.141501	5.141501	5.141501	5.141501	5.141501	5.141501	5.141501	5.141501	5.141501	5.141501	5.141501	5.141501	5.141501	5.141501
0.65	5.141501	5.141501	5.141501	5.141501	5.141501	5.141501	5.141501	5.141501	5.141501	5.141501	5.141501	5.141501	5.141501	5.141501	5.141501	5.141501	5.141501	5.141501	5.141501	5.141501	2.53553	5.141501	5.141501	5.141501	5.141501	5.141501	5.141501	5.141501	5.141501
0.75	5.141501	5.141501	5.141501	5.141501	5.141501	5.141501	5.141501	5.141501	5.141501	5.141501	5.141501	5.141501	5.141501	5.141501	5.141501	5.141501	5.141501	5.141501	5.141501	5.141501	5.141501	5.141501	5.141501	5.141501	5.141501	5.141501	5.141501	5.141501	5.141501
0.85	5.141501	5.141501	5.141501	5.141501	5.141501	5.141501	5.141501	5.141501	5.141501	5.141501	5.141501	5.141501	5.141501	5.141501	5.141501	5.141501	5.141501	5.141501	5.141501	5.141501	5.141501	5.141501	5.141501	5.141501	5.141501	5.141501	5.141501	5.141501	5.141501
0.95	5.141501	5.141501	5.141501	5.141501	5.141501	5.141501	5.141501	5.141501	5.141501	5.141501	5.141501	5.141501	5.141501	5.141501	5.141501	5.141501	5.141501	5.141501	5.141501	5.141501	5.141501	5.141501	5.141501	5.141501	5.141501	5.141501	5.141501	5.141501	5.141501
1.05	3.382393	5.141501	5.141501	5.141501	5.141501	3.382393	5.141501	3.382393	5.141501	5.141501	5.141501	5.141501	5.141501	5.141501	5.141501	5.141501	3.382393	5.141501	5.141501	5.141501	5.141501	5.141501	5.141501	5.141501	5.141501	5.141501	5.141501	5.141501	5.141501
1.15	3.530552	5.141501	5.141501	3.530552	5.141501	5.141501	5.141501	5.141501	5.141501	5.141501	5.141501	5.141501	5.141501	5.141501	5.141501	5.141501	5.141501	5.141501	5.141501	5.141501	5.141501	5.141501	5.141501	5.141501	5.141501	5.141501	5.141501	5.141501	5.141501
1.25	3.661717	3.089431	3.661717	5.141501	5.141501	5.141501	5.141501	5.141501	5.141501	5.141501	5.141501	5.141501	5.141501	5.141501	5.141501	5.141501	5.141501	5.141501	5.141501	5.141501	5.141501	5.141501	5.141501	5.141501	5.141501	5.141501	5.141501	5.141501	5.141501
1.35	2.866876	3.778452	3.778452	3.778452	5.141501	5.141501	5.141501	5.141501	5.141501	5.141501	5.141501	5.141501	5.141501	5.141501	5.141501	5.141501	5.141501	5.141501	5.141501	5.141501	5.141501	5.141501	5.141501	5.141501	5.141501	5.141501	5.141501	5.141501	5.141501
1.45	1.621913	2.531134	2.215888	3.34283	3.882811	2.994118	5.141501	3.882811	5.141501	5.141501	5.141501	5.141501	5.141501	5.141501	5.141501	5.141501	5.141501	5.141501	5.141501	5.141501	5.141501	5.141501	5.141501	5.141501	5.141501	5.141501	5.141501	5.141501	5.141501
1.55	1.612482	2.215747	3.110974	2.654083	3.976466	3.45287	3.976466	5.141501	5.141501	5.141501	5.141501	3.976466	5.141501	5.141501	5.141501	5.141501	5.141501	3.976466	5.141501	5.141501	5.141501	5.141501	5.141501	5.141501	5.141501	5.141501	5.141501	5.141501	5.141501
1.65	-0.10465	0.85064	1.23613	2.122668	2.031809	2.333671	2.45862	5.141501	5.141501	4.060802	4.060802	4.060802	3.553598	5.141501	3.553598	5.141501	4.060802	5.141501	5.141501	5.141501	5.141501	5.141501	5.141501	5.141501	5.141501	5.141501	5.141501	5.141501	5.141501
1.75	-1.40734	-0.47322	0.322241	1.140571	1.568812	2.234266	2.874226	3.31827	3.646078	2.709321	3.31827	3.646078	5.141501	3.31827	5.141501	4.136976	4.136976	5.141501	5.141501	5.141501	5.141501	5.141501	5.141501	5.141501	5.141501	5.141501	5.141501	5.141501	5.141501
1.85	-2.44075	-1.63909	-0.70142	-0.19103	0.289602	0.649849	1.281672	1.953966	2.810188	2.437648	4.205967	3.410603	4.205967	4.205967	4.205967	4.205967	3.410603	3.73121	5.141501	5.141501	5.141501	5.141501	4.205967	5.141501	5.141501	4.205967	5.141501	5.141501	5.141501
1.95	-3.04326	-2.28372	-1.37411	-0.856	-0.47739	0.187254	0.654995	1.153163	1.291438	2.266478	2.120419	2.765969	3.258184	3.258184	3.496415	3.496415	3.809758	5.141501	5.141501	5.141501	5.141501	5.141501	3.809758	4.268611	5.141501	5.141501	5.141501	5.141501	5.141501
2.05	-3.19184	-2.50738	-1.74441	-1.28858	-0.8232	-0.61116	-0.13261	0.517585	0.905913	1.085345	1.827889	1.698916	2.993593	2.993593	2.441974	3.152837	3.342333	3.882385	4.325626	4.325626	3.882385	5.141501	5.141501	5.141501	5.141501	5.141501	5.141501	5.141501	5.141501
2.15	-3.00093	-2.42649	-1.69707	-1.38362	-1.04706	-0.82089	-0.61158	-0.28289	-0.00505	0.688948	0.772478	1.453041	1.919515	2.179065	3.07725	2.240163	3.234506	3.650915	3.234506	3.421191	3.650915	4.377632	4.377632	5.141501	3.949663	4.377632	5.141501	5.141501	5.141501
2.25	-2.60054	-2.00763	-1.50823	-1.19583	-1.01793	-0.92526	-0.69417	-0.57062	-0.36799	-0.14055	0.098746	0.618828	0.876807	1.289989	2.153577	1.91947	2.153577	3.156167	2.61449	3.021831	3.720627	5.141501	4.012093	5.141501	3.720627	4.012093	5.141501	5.141501	5.141501
2.35	-2.34034	-1.73016	-1.26633	-1.1436	-0.90424	-0.87426	-0.64054	-0.57325	-0.5669	-0.41595	-0.3188	-0.01056	0.158469	0.628307	0.93353	1.543935	2.002824	1.78003	2.47178	2.235979	3.383893	3.230736	3.78589	5.141501	3.383893	5.141501	5.141501	5.141501	5.141501
2.45	-2.15827	-1.40876	-1.12951	-0.90601	-0.74807	-0.57796	-0.60652	-0.66397	-0.58374	-0.54257	-0.46456	-0.31802	-0.077	0.077845	0.352342	0.591904	1.312181	1.52515	2.041807	1.651477	2.168921	2.690877	3.45236	3.301302	3.630369	3.630369	5.141501	5.141501	5.141501
2.55	-1.9911	-1.18231	-0.8345	-0.61303	-0.53239	-0.49822	-0.34514	-0.54629	-0.43546	-0.43854	-0.40095	-0.30379	-0.33837	-0.22703	-0.07113	0.25084	0.367684	0.86607	1.316586	1.409717	2.005706	1.904702	2.49951	2.690848	3.692166	3.904516	5.141501	5.141501	5.141501
2.65	-1.88986	-1.01696	-0.69149	-0.41867	-0.36498	-0.28859	-0.22014	-0.173	-0.33069	-0.24117	-0.31148	-0.32751	-0.20228	-0.14285	-0.1618	-0.12743	0.018774	0.24493	0.515235	0.855186	1.243539	1.751053	2.079169	2.570763	2.570763	4.579194	5.141501	5.141501	5.141501
2.75	-1.76312	-0.8906	-0.43903	-0.26953	-0.28506	-0.17091	-0.11404	0.01043	-0.00192	-0.11037	-0.06533	-0.06148	-0.08057	-0.20153	-0.15699	-0.16745	-0.06148	0.096904	0.143117	0.566654	0.748359	1.061111	1.379149	1.874607	2.149661	3.805631	5.141501	5.141501	5.141501
2.85	-1.60746	-0.58967	-0.31359	-0.19529	0.001874	0.052859	0.128041	0.073182	0.132388	0.223554	0.167863	0.223554	0.013411	0.009551	-0.00194	0.04086	0.119402	0.089743	0.181498	0.257532	0.389279	0.881383	0.928547	1.143196	1.818163	4.306413	5.141501	5.141501	5.141501
2.95	-1.21446	-0.27242	-0.14839	0.078101	0.162441	0.214033	0.366243	0.46328	0.282509	0.376572	0.350948	0.418986	0.509627	0.452021	0.424417	0.413585	0.273103	0.325964	0.452021	0.371394	0.480409	0.564471	0.757319	1.33819	1.394037	5.141501	5.141501	5.141501	5.141501
3.05	-0.65853	-0.0317	0.175621	0.254091	0.422118	0.437482	0.422118	0.688421	0.675241	0.771367	0.681809	0.662233	0.715312	0.838428	0.474279	0.649393	0.617994	0.617994	0.808067	0.757056	0.735966	0.910313	1.071728	1.195912	1.533464	5.141501	5.141501	5.141501	5.141501
3.15	-0.19208	0.451106	0.565295	0.675634	0.651371	0.633553	0.785869	0.772251	0.792748	0.973866	0.998999	0.941309	0.925419	1.02478	0.973866	1.007519	1.051243	0.917568	0.973866	1.125439	0.990551	0.949349	1.060222	1.164708	2.082921	5.141501	5.141501	5.141501	5.141501
3.25	0.309672	0.933318	1.052041	1.002847	1.068991	0.925882	0.911175	0.940809	0.971347	1.256467	1.236056	1.068991	1.216054	1.226005	1.167732	1.216054	1.186782	1.320323	1.331374	1.298581	1.206201	1.309393	1.720347	1.287885	2.464053	5.141501	5.141501	5.141501	5.141501
3.35	0.714902	1.246196	1.447683	1.137113	1.326144	1.078577	1.236637	1.447683	1.128538	1.246196	1.40176	1.326144	1.483565	1.572662	1.495817	1.471461	1.255847	1.533502	1.627383	1.572662	1.483565	1.846606	1.938215	1.700283	2.751127	5.141501	5.141501	5.141501	5.141501
3.45	1.112313	1.684408	1.656723	1.787827	1.481814	1.448007	1.212191	1.404637	1.553065	1.577989	1.540831	1.684408	1.616581	1.819453	1.803515	1.772382	1.803515	1.975625	1.819453	2.138434	1.835649	1.868849	2.05372	1.852111	3.162711	5.141501	5.141501	5.141501	5.141501
3.55	1.537448	1.827507	1.940754	1.923769	1.842921	1.646149	1.842921	1.671997	1.890641	1.712068	1.812327	1.712068	1.940754	1.608587	1.739703	1.827507	2.068483	1.993511	1.890641	1.958032	2.049206	2.088139	2.284945	1.858576	4.00921	5.141501	5.141501	5.141501	5.141501
3.65	1.568858	2.313423	1.927864	2.083312	1.943989	2.102175	1.911994	2.028766	1.725779	1.96038	1.927864	2.028766	2.04662	2.202231	1.993988	2.064797	2.064797	2.361609	2.011226	2.141005	2.337226	2.290174	2.386602	2.645074	5.141501	5.141501	5.141501	5.141501	5.141501
3.75	2.012132	2.412226	2.153577	2.515803	2.364202	1.99578	2.134763	2.062859	2.062859	2.028757	2.172752	2.212241	2.172752	2.134763	2.172752	2.232587	2.341027	2.341027	2.63136	2.762038	2.437132	2.488886	2.601195	2.662464	5.141501	5.141501	5.141501	5.141501	5.141501
3.85	2.302984	2.390382	2.564533	2.41348	2.222615	2.437124	2.302984	2.130463	2.222615	2.095855	2.014239	2.302984	2.261992	2.390382	2.437124	2.302984	2.345729	2.592085	2.592085	2.537719	2.564533	2.80968	2.710578	3.290928	5.141501	5.141501	5.141501	5.141501	5.141501
3.95	2.196877	2.585118	2.533751	2.92747	2.757246	2.415811	2.611826	2.310326	2.351195	2.484894	2.271062	2.233282	2.509024	2.559104	2.509024	2.484894	2.372273	2.585118	2.965291	2.757246	2.821901	2.757246	2.639267	3.279746	5.141501	5.141501	5.141501	5.141501	5.141501

Figure 5: Acceptor-Orbital-Hydrogen Angles for π - π interactions represented as a heatmap.

For π - π interactions, one of the two atoms is considered to be the donor and the other is the acceptor even though it is more strictly just an interaction. The interaction map shows that the interactions happen most frequently at a distance of 2 to 2.1 angstroms and at around 180^0. Because there is no real difference from one end to the other, they look identical. π - π interactions include parallel π-π interactions and cation-π interactions.

	-0.975	-0.925	-0.875	-0.825	-0.775	-0.725	-0.675	-0.625	-0.575	-0.525	-0.475	-0.425	-0.375	-0.325	-0.275	-0.225	-0.175	-0.125	-0.075	-0.025	0.025	0.075	0.125	0.175	0.225	0.275	0.325	0.375
0.05	5.192397	5.192397	5.192397	5.192397	5.192397	5.192397	5.192397	5.192397	5.192397	5.192397	5.192397	5.192397	5.192397	5.192397	5.192397	5.192397	5.192397	5.192397	5.192397	5.192397	5.192397	5.192397	5.192397	5.192397	5.192397	5.192397	5.192397	5.192397
0.15	5.192397	5.192397	5.192397	5.192397	5.192397	5.192397	5.192397	5.192397	5.192397	5.192397	5.192397	5.192397	5.192397	5.192397	5.192397	5.192397	5.192397	5.192397	5.192397	5.192397	5.192397	5.192397	5.192397	5.192397	5.192397	5.192397	5.192397	5.192397
0.25	5.192397	5.192397	0.740378	5.192397	5.192397	5.192397	5.192397	5.192397	5.192397	5.192397	5.192397	5.192397	5.192397	5.192397	5.192397	5.192397	5.192397	5.192397	5.192397	5.192397	5.192397	5.192397	5.192397	5.192397	5.192397	5.192397	5.192397	5.192397
0.35	5.192397	5.192397	5.192397	5.192397	1.402196	5.192397	5.192397	5.192397	5.192397	5.192397	5.192397	5.192397	5.192397	5.192397	5.192397	5.192397	5.192397	5.192397	5.192397	5.192397	5.192397	5.192397	5.192397	5.192397	5.192397	5.192397	5.192397	5.192397
0.45	5.192397	5.192397	5.192397	1.890179	5.192397	5.192397	5.192397	5.192397	5.192397	5.192397	5.192397	5.192397	5.192397	5.192397	5.192397	5.192397	5.192397	5.192397	5.192397	5.192397	5.192397	5.192397	5.192397	5.192397	5.192397	5.192397	5.192397	5.192397
0.55	5.192397	5.192397	5.192397	5.192397	5.192397	5.192397	5.192397	5.192397	5.192397	5.192397	5.192397	2.27351	5.192397	5.192397	5.192397	5.192397	5.192397	5.192397	5.192397	5.192397	5.192397	5.192397	5.192397	5.192397	5.192397	5.192397	5.192397	5.192397
0.65	5.192397	5.192397	5.192397	5.192397	5.192397	5.192397	5.192397	5.192397	2.586426	5.192397	5.192397	5.192397	5.192397	5.192397	5.192397	5.192397	5.192397	5.192397	5.192397	5.192397	5.192397	5.192397	5.192397	5.192397	5.192397	5.192397	5.192397	5.192397
0.75	5.192397	5.192397	5.192397	5.192397	5.192397	5.192397	5.192397	5.192397	5.192397	5.192397	5.192397	5.192397	5.192397	5.192397	5.192397	5.192397	5.192397	5.192397	5.192397	5.192397	5.192397	5.192397	5.192397	5.192397	5.192397	2.848457	2.848457	5.192397
0.85	5.192397	5.192397	5.192397	5.192397	5.192397	5.192397	5.192397	3.071857	5.192397	5.192397	5.192397	5.192397	5.192397	5.192397	5.192397	5.192397	5.192397	5.192397	5.192397	5.192397	5.192397	5.192397	5.192397	5.192397	5.192397	5.192397	5.192397	5.192397
0.95	3.264858	3.264858	5.192397	5.192397	5.192397	5.192397	5.192397	5.192397	5.192397	5.192397	5.192397	3.264858	5.192397	5.192397	5.192397	5.192397	5.192397	3.264858	5.192397	5.192397	5.192397	3.264858	5.192397	5.192397	5.192397	5.192397	5.192397	5.192397
1.05	3.433289	5.192397	5.192397	2.830175	5.192397	2.830175	3.433289	5.192397	5.192397	5.192397	5.192397	5.192397	3.433289	5.192397	3.433289	5.192397	5.192397	5.192397	5.192397	5.192397	5.192397	5.192397	5.192397	5.192397	5.192397	5.192397	5.192397	5.192397
1.15	3.581448	3.581448	5.192397	3.581448	2.993494	5.192397	2.993494	5.192397	5.192397	3.581448	5.192397	5.192397	5.192397	5.192397	2.993494	5.192397	5.192397	3.581448	5.192397	5.192397	5.192397	5.192397	5.192397	5.192397	5.192397	5.192397	5.192397	5.192397
1.25	2.131252	2.51357	2.30425	2.51357	5.192397	3.140327	5.192397	3.712613	5.192397	5.192397	3.712613	3.712613	5.192397	5.192397	5.192397	5.192397	5.192397	5.192397	5.192397	5.192397	5.192397	5.192397	5.192397	5.192397	5.192397	5.192397	5.192397	5.192397
1.35	1.394184	1.61174	1.461577	1.788506	2.277411	2.44895	3.829348	3.273097	3.829348	3.273097	5.192397	5.192397	3.829348	5.192397	5.192397	5.192397	5.192397	5.192397	5.192397	5.192397	3.829348	5.192397	5.192397	5.192397	5.192397	5.192397	5.192397	5.192397
1.45	0.013671	0.347892	0.520747	1.158904	2.58203	1.533664	2.139979	2.786964	2.412036	3.933707	2.786964	3.933707	3.933707	3.393726	3.393726	5.192397	5.192397	5.192397	5.192397	5.192397	5.192397	5.192397	5.192397	5.192397	5.192397	5.192397	5.192397	5.192397
1.55	-0.50743	-0.18838	0.282923	0.422392	0.861595	1.204435	1.730277	1.879214	1.879214	2.154836	3.161869	2.266643	4.027362	4.027362	4.027362	4.027362	5.192397	4.027362	4.027362	5.192397	5.192397	5.192397	4.027362	4.027362	5.192397	5.192397	5.192397	5.192397
1.65	-1.19248	-0.88323	-0.40473	0.077565	0.201927	0.665223	0.824601	1.24858	1.108066	1.922539	2.652337	2.652337	2.384567	2.819009	3.019128	3.269577	4.111698	4.111698	5.192397	5.192397	5.192397	5.192397	4.111698	5.192397	5.192397	4.111698	4.111698	5.192397
1.75	-1.53107	-1.31082	-1.00356	-0.70593	-0.64726	-0.22805	-0.05271	0.52362	0.991244	1.130056	1.669524	1.442077	2.035484	2.494741	2.285162	3.122697	3.369165	3.369165	3.369165	5.192397	5.192397	4.187871	5.192397	4.187871	5.192397	5.192397	5.192397	5.192397
1.85	-1.95259	-1.68372	-1.53585	-1.32863	-1.20245	-0.96409	-0.77296	-0.42473	0.00453	0.114083	0.601729	0.91248	1.300458	1.400056	2.21774	2.30012	3.782105	3.219108	3.219108	2.720911	5.192397	4.256863	3.782105	5.192397	4.256863	5.192397	5.192397	5.192397
1.95	-2.10352	-1.94838	-1.82141	-1.71154	-1.50564	-1.39294	-1.24158	-1.06047	-0.777	-0.59803	-0.27966	0.042349	0.432902	0.690318	0.822267	1.230226	2.171315	1.985799	2.816865	2.586426	4.319507	4.319507	3.30908	5.192397	5.192397	5.192397	5.192397	5.192397
2.05	-2.06853	-1.89587	-1.7858	-1.7799	-1.7114	-1.62137	-1.49764	-1.35099	-1.2697	-0.97926	-0.86502	-0.52771	-0.1683	-0.008	0.327653	0.496845	0.993993	1.470157	1.635589	2.67871	2.411478	3.393229	2.67871	2.907154	3.627233	3.627233	3.933281	4.376522
2.15	-1.81947	-1.6871	-1.70811	-1.52761	-1.59319	-1.57489	-1.56827	-1.44397	-1.38671	-1.2394	-1.10952	-0.90505	-0.76704	-0.4733	-0.29563	-0.11413	0.260893	0.546172	0.914522	0.99739	1.692748	2.172381	2.872707	2.765906	3.701811	3.472087	5.192397	4.428528
2.25	-1.40589	-1.39297	-1.39441	-1.41583	-1.40304	-1.4478	-1.38572	-1.30858	-1.33342	-1.25197	-1.1651	-1.04776	-0.94012	-0.69396	-0.53688	-0.44799	-0.23314	-0.06814	0.342897	0.942527	0.765085	1.409949	1.852375	2.510699	4.062988	3.207063	4.062988	4.476064
2.35	-1.03944	-1.0801	-1.05998	-1.05442	-1.20909	-1.12092	-1.20749	-1.18654	-1.10691	-1.12785	-1.12613	-0.90861	-0.93618	-0.87137	-0.81158	-0.57775	-0.62453	-0.46706	-0.28811	0.029284	0.196296	0.627931	1.189734	1.340844	2.59147	2.831817	2.926747	3.148846
2.45	-0.74023	-0.77474	-0.82115	-0.902	-0.95815	-0.94488	-0.99873	-0.93143	-0.96938	-1.00054	-0.97124	-0.83402	-0.88391	-0.77474	-0.78151	-0.70932	-0.67493	-0.73082	-0.59158	-0.4169	-0.20512	0.112132	0.360106	0.920782	1.402196	1.81635	2.314529	3.105022
2.55	-0.39683	-0.49815	-0.53862	-0.56985	-0.70842	-0.64392	-0.74565	-0.77738	-0.86314	-0.88207	-0.78361	-0.67207	-0.64629	-0.76692	-0.80409	-0.70619	-0.73701	-0.69946	-0.63435	-0.63675	-0.54652	-0.3372	-0.23515	0.092444	0.527599	1.282291	1.779846	2.209414
2.65	-0.15857	-0.11839	-0.2924	-0.43516	-0.46458	-0.57904	-0.58374	-0.63171	-0.60458	-0.69019	-0.61143	-0.59305	-0.68808	-0.72126	-0.56722	-0.72533	-0.6494	-0.548	-0.52841	-0.58139	-0.56006	-0.52345	-0.31714	-0.1405	0.004379	0.700375	1.100256	1.535632
2.75	0.082253	0.116676	-0.15063	-0.22488	-0.23417	-0.42191	-0.47174	-0.3775	-0.51689	-0.38813	-0.46442	-0.52383	-0.49338	-0.52614	-0.57334	-0.62886	-0.57554	-0.59291	-0.57554	-0.51921	-0.56008	-0.40905	-0.46931	-0.3084	-0.04469	0.298624	0.561366	1.148187
2.85	0.333425	0.218759	0.015246	-0.16613	-0.05239	-0.11896	-0.21995	-0.3327	-0.2902	-0.32224	-0.28748	-0.36595	-0.28748	-0.42457	-0.48442	-0.45724	-0.37097	-0.37097	-0.41023	-0.43636	-0.4387	-0.37347	-0.3834	-0.24581	-0.09945	0.037591	0.260233	0.710117
2.95	0.514176	0.357313	0.256143	0.209155	0.140619	0.009351	-0.09134	-0.12476	-0.12476	-0.0975	-0.14253	-0.22963	-0.17998	-0.2692	-0.30227	-0.25618	-0.29473	-0.33911	-0.25356	-0.11274	-0.22423	-0.22694	-0.19681	-0.19681	-0.10056	0.065577	0.296302	0.627974
3.05	0.773045	0.503983	0.638447	0.457881	0.428287	0.068828	0.128233	0.075637	-0.03729	0.062066	-0.15244	-0.10796	-0.18192	-0.18718	-0.1713	-0.09076	-0.1108	-0.14972	-0.14699	-0.05843	-0.09653	6.42E-06	0.009553	0.015968	0.062066	0.26292	0.357919	0.503983
3.15	0.968464	0.789893	0.816407	0.610708	0.578425	0.399526	0.32696	0.170971	0.129568	0.129568	0.294552	0.11281	-0.01786	0.039146	-0.01494	0.093064	0.048453	0.080114	0.017763	0.116139	0.232726	0.188742	0.214163	0.210492	0.377631	0.40842	0.347762	0.547152
3.25	1.128471	1.102936	0.912236	0.83219	0.740378	0.645596	0.534927	0.435294	0.344694	0.418199	0.344694	0.212144	0.212144	0.236579	0.158438	0.236579	0.254405	0.317507	0.208701	0.306078	0.205271	0.276224	0.340765	0.435294	0.443952	0.544496	0.470386	0.635014
3.35	1.268685	1.089664	1.121383	0.924554	0.872968	0.77146	0.77146	0.508376	0.609325	0.521557	0.474056	0.432747	0.469847	0.424687	0.525989	0.623916	0.275778	0.428709	0.432747	0.457327	0.553003	0.449066	0.358673	0.54845	0.700243	0.609325	0.812018	1.014507
3.45	1.317035	1.094345	1.211901	1.109244	1.050938	0.950009	0.894005	0.74788	0.72718	0.687023	0.545068	0.545068	0.774372	0.611246	0.536566	0.532342	0.687023	0.593166	0.796083	0.615817	0.657929	0.483	0.495109	0.677231	0.732315	0.858339	0.950009	1.124367
3.55	1.46977	1.532699	1.310373	1.293188	1.072394	1.025681	1.114248	0.974837	0.987308	0.914715	0.83619	0.748449	0.83081	0.863532	0.753403	0.938332	0.83619	0.841599	0.83619	0.999936	0.83619	0.920567	0.820136	0.804336	0.944324	1.188149	1.128598	1.391588
3.65	1.750872	1.464827	1.455304	1.474441	1.265921	1.219941	1.048304	1.073862	1.154707	0.918338	0.987128	0.92948	0.848797	0.963668	1.048304	1.061001	1.054632	0.975329	1.183176	1.029555	1.048304	0.940746	0.935097	0.963668	1.127026	1.154707	1.161748	1.39107
3.75	1.57659	1.741521	1.508039	1.408996	1.35072	1.272953	1.243464	1.101481	1.088953	1.200799	1.046304	1.034445	1.127016	1.173335	1.153221	1.334676	1.318884	1.200799	1.35884	1.408996	1.2581	1.2581	1.28803	1.159881	1.146605	1.586779	1.452804	1.617981
3.85	1.961024	1.559317	1.669102	1.435041	1.339576	1.394151	1.354867	1.280648	1.339576	1.49525	1.146934	1.295057	1.324515	1.410307	1.443423	1.578403	1.578403	1.627773	1.578403	1.588083	1.669102	1.38617	1.617702	1.578403	1.49525	1.568814	1.70126	1.617702
3.95	1.830131	1.729424	1.572152	1.563097	1.59051	1.452293	1.295775	1.31669	1.412744	1.485098	1.35255	1.345275	1.657557	1.554124	1.599817	1.609212	1.647699	1.677569	1.667513	1.628271	1.618696	1.772936	1.687727	1.853957	1.982431	1.729424	1.853957	2.024429

Figure 6: Donor-Hydrogen-Orbital Angles for π -hydrogen interactions represented as a heatmap.

	-0.975	-0.925	-0.875	-0.825	-0.775	-0.725	-0.675	-0.625	-0.575	-0.525	-0.475	-0.425	-0.375	-0.325	-0.275	-0.225	-0.175	-0.125	-0.075	-0.025	0.025	0.075	0.125	0.175	0.225	0.275	0.325	0.375
0.05	5.739646	5.739646	5.739646	5.739646	5.739646	5.739646	5.739646	5.739646	5.739646	5.739646	5.739646	5.739646	5.739646	5.739646	5.739646	5.739646	5.739646	5.739646	5.739646	5.739646	5.739646	5.739646	5.739646	5.739646	5.739646	5.739646	5.739646	5.739646
0.15	5.739646	5.739646	5.739646	5.739646	5.739646	5.739646	5.739646	5.739646	5.739646	5.739646	5.739646	5.739646	5.739646	5.739646	5.739646	5.739646	5.739646	5.739646	5.739646	5.739646	5.739646	5.739646	5.739646	5.739646	5.739646	5.739646	5.739646	5.739646
0.25	5.739646	5.739646	5.739646	5.739646	5.739646	5.739646	5.739646	5.739646	5.739646	5.739646	5.739646	5.739646	5.739646	5.739646	5.739646	5.739646	5.739646	5.739646	5.739646	5.739646	5.739646	5.739646	5.739646	5.739646	5.739646	5.739646	5.739646	5.739646
0.35	5.739646	5.739646	5.739646	5.739646	5.739646	5.739646	5.739646	5.739646	5.739646	1.267657	5.739646	5.739646	5.739646	5.739646	5.739646	5.739646	5.739646	5.739646	5.739646	5.739646	5.739646	5.739646	5.739646	5.739646	5.739646	5.739646	5.739646	5.739646
0.45	5.739646	5.739646	5.739646	5.739646	5.739646	5.739646	5.739646	5.739646	5.739646	5.739646	5.739646	5.739646	5.739646	5.739646	5.739646	5.739646	5.739646	5.739646	5.739646	5.739646	5.739646	5.739646	1.762853	5.739646	5.739646	5.739646	5.739646	5.739646
0.55	5.739646	5.739646	5.739646	5.739646	5.739646	5.739646	5.739646	5.739646	5.739646	5.739646	5.739646	5.739646	5.739646	5.739646	5.739646	5.739646	5.739646	5.739646	5.739646	5.739646	5.739646	5.739646	5.739646	5.739646	5.739646	5.739646	5.739646	5.739646
0.65	5.739646	5.739646	5.739646	5.739646	5.739646	5.739646	5.739646	5.739646	5.739646	5.739646	5.739646	5.739646	5.739646	5.739646	5.739646	5.739646	5.739646	5.739646	5.739646	5.739646	5.739646	5.739646	5.739646	5.739646	5.739646	5.739646	5.739646	5.739646
0.75	5.739646	5.739646	5.739646	5.739646	5.739646	5.739646	2.751722	5.739646	5.739646	5.739646	5.739646	5.739646	5.739646	5.739646	5.739646	3.395705	5.739646	5.739646	5.739646	5.739646	5.739646	5.739646	3.395705	5.739646	5.739646	5.739646	5.739646	5.739646
0.85	5.739646	5.739646	5.739646	5.739646	5.739646	5.739646	5.739646	5.739646	5.739646	5.739646	5.739646	5.739646	5.739646	5.739646	5.739646	5.739646	5.739646	5.739646	5.739646	5.739646	5.739646	5.739646	5.739646	5.739646	5.739646	5.739646	5.739646	2.987816
0.95	5.739646	5.739646	5.739646	3.194495	5.739646	5.739646	5.739646	5.739646	5.739646	3.194495	5.739646	5.739646	5.739646	5.739646	5.739646	3.812107	5.739646	5.739646	3.812107	5.739646	5.739646	5.739646	5.739646	5.739646	3.812107	5.739646	5.739646	5.739646
1.05	2.732527	5.739646	5.739646	3.377424	3.377424	3.980538	5.739646	5.739646	3.377424	5.739646	3.377424	5.739646	5.739646	5.739646	5.739646	5.739646	5.739646	5.739646	5.739646	5.739646	5.739646	5.739646	5.739646	5.739646	5.739646	5.739646	5.739646	5.739646
1.15	3.172953	5.739646	2.693325	2.370526	5.739646	5.739646	5.739646	5.739646	3.540742	5.739646	5.739646	5.739646	5.739646	3.540742	5.739646	5.739646	5.739646	5.739646	5.739646	5.739646	5.739646	5.739646	5.739646	5.739646	5.739646	5.739646	5.739646	5.739646
1.25	2.186586	2.402596	2.402596	2.6785	3.687575	3.325877	5.739646	3.325877	3.687575	5.739646	5.739646	3.060818	5.739646	5.739646	5.739646	5.739646	5.739646	5.739646	5.739646	5.739646	5.739646	5.739646	5.739646	5.739646	5.739646	5.739646	5.739646	5.739646
1.35	1.079123	1.565596	1.762853	2.824659	2.550621	2.008826	2.158989	2.824659	3.820346	3.820346	2.824659	3.820346	5.739646	5.739646	3.820346	5.739646	5.739646	5.739646	5.739646	3.820346	5.739646	5.739646	5.739646	5.739646	5.739646	5.739646	5.739646	5.739646
1.45	0.041025	0.084174	1.021901	1.193484	1.400732	1.662471	2.148067	2.148067	2.959284	2.017986	2.687227	2.297636	5.739646	5.739646	3.940974	5.739646	5.739646	5.739646	5.739646	5.739646	5.739646	3.940974	5.739646	5.739646	5.739646	4.480955	5.739646	5.739646
1.55	-0.65534	-0.33198	0.190958	0.798145	1.131763	1.565643	1.352417	1.408844	2.277526	1.600185	1.600185	3.083857	4.051014	4.051014	4.051014	5.739646	5.739646	4.051014	4.051014	5.739646	5.739646	5.739646	5.739646	5.739646	5.739646	5.739646	5.739646	5.739646
1.65	-1.33753	-0.97745	-0.23237	0.386613	0.73602	0.817683	0.987283	1.300414	1.300414	1.004229	2.2693	2.331767	4.151742	3.199585	4.151742	3.199585	3.566377	3.816826	3.199585	5.739646	4.151742	5.739646	5.739646	4.658946	5.739646	4.151742	5.739646	5.739646
1.75	-1.84549	-1.47906	-0.80672	-0.24071	0.259738	0.652377	0.570345	1.173386	1.210035	1.267657	1.677304	1.874158	3.041989	2.659235	3.669945	3.669945	5.739646	4.73512	5.739646	3.307466	5.739646	4.244222	5.739646	4.73512	5.739646	5.739646	5.739646	5.739646
1.85	-2.31969	-1.96217	-1.23676	-0.68576	-0.06668	0.534632	0.595976	0.840482	1.003044	0.89992	1.573111	1.879816	2.2748	1.982838	2.937149	3.408332	5.739646	3.766356	3.408332	5.739646	4.804111	4.329354	2.937149	4.329354	5.739646	5.739646	5.739646	5.739646
1.95	-2.51347	-2.22078	-1.66289	-1.12868	-0.4027	-0.24038	0.156805	0.412296	0.856699	0.899848	1.134864	1.317973	1.919689	2.085535	2.284464	2.946518	2.376621	2.718564	3.133674	3.502889	3.856329	5.739646	3.856329	3.502889	3.856329	4.407903	4.866755	5.739646
2.05	-2.53437	-2.31554	-1.82527	-1.37255	-0.82173	-0.38897	-0.14063	0.156742	0.446335	0.596487	0.764489	1.140795	1.433597	1.306021	1.849381	1.875497	2.426033	2.813469	3.128728	3.333671	3.591737	4.480529	3.454402	4.92377	4.174481	3.940478	3.591737	5.739646
2.15	-2.48994	-2.23226	-1.83426	-1.44017	-0.96502	-0.70134	-0.44183	-0.12356	0.135354	0.310212	0.841794	0.783624	1.341997	1.314168	1.616261	2.472954	1.75545	2.613554	2.349707	2.665186	2.71963	4.019335	3.047807	3.216668	4.24906	3.419955	4.547807	5.739646
2.25	-2.46756	-2.01009	-1.71652	-1.29273	-0.90465	-0.73746	-0.57899	-0.23301	-0.1202	0.269047	0.346905	0.618906	0.975701	1.161682	1.58366	1.844606	1.617022	1.669248	2.517614	2.32805	2.476718	3.501566	2.70031	3.212634	2.517614	3.057948	5.739646	5.739646
2.35	-2.3645	-1.8718	-1.45776	-1.19188	-0.9503	-0.78455	-0.5579	-0.39726	-0.16476	-0.05124	0.14583	0.35124	0.604383	0.936252	0.936252	1.325934	1.560645	1.423523	1.867907	1.929729	2.345429	2.313723	2.687821	3.00556	2.94509	3.069925	4.16297	5.739646
2.45	-2.30043	-1.72073	-1.31377	-1.01229	-0.83432	-0.74676	-0.66111	-0.52467	-0.4109	-0.18829	-0.02513	0.33416	0.295265	0.575179	0.845958	1.16243	1.050854	1.396684	1.751071	1.657446	1.854375	2.19716	2.099837	2.032596	2.19716	2.458509	5.739646	5.739646
2.55	-2.23064	-1.5811	-1.21612	-0.99922	-0.80138	-0.62847	-0.63127	-0.52548	-0.35524	-0.23429	-0.18324	0.051855	0.232948	0.374916	0.509413	0.624894	1.074847	1.113998	1.419874	1.510611	2.068209	2.382316	2.027575	2.110564	1.950949	2.987816	5.739646	5.739646
2.65	-2.06021	-1.47965	-1.1093	-0.87394	-0.72456	-0.64885	-0.50151	-0.45333	-0.35282	-0.37808	-0.18214	-0.12168	-0.008	0.090605	0.39948	0.534914	0.741287	1.021652	1.035228	1.166492	1.574591	1.796782	1.841683	1.872778	2.122627	2.829554	5.739646	5.739646
2.75	-1.92691	-1.32629	-0.96998	-0.7795	-0.64706	-0.53479	-0.48053	-0.36073	-0.34009	-0.28231	-0.1823	-0.13815	0.005128	-0.00613	0.273792	0.344389	0.650877	0.575975	0.746002	1.075038	1.312436	1.505827	1.812563	1.812563	1.446654	3.359029	5.739646	5.739646
2.85	-1.63093	-1.12135	-0.77125	-0.60023	-0.51745	-0.42308	-0.37954	-0.34892	-0.264	-0.24118	-0.24447	-0.10163	-0.06298	4.09E-05	0.161376	0.161376	0.412315	0.631085	0.512221	0.906312	0.836119	1.34201	1.08808	1.434487	1.526691	4.28978	5.739646	5.739646
2.95	-1.28828	-0.85673	-0.56867	-0.3432	-0.40124	-0.31645	-0.19842	-0.10241	-0.13545	-0.13031	-0.03854	0.001841	-0.12686	0.081452	0.154148	0.127098	0.320313	0.506613	0.570362	0.598469	0.672401	0.777488	0.904562	1.044583	1.220643	5.26402	5.739646	5.739646
3.05	-0.96023	-0.4521	-0.27391	-0.2753	-0.18661	-0.13514	-0.11574	-0.0324	0.003723	0.08411	0.098159	0.035486	0.05079	-0.04474	-0.00178	0.147884	0.248358	0.378621	0.394805	0.51915	0.513011	0.73864	0.975535	0.742482	1.516773	5.739646	5.739646	5.739646
3.15	-0.59822	-0.14183	-0.05304	0.079123	0.062529	0.293994	0.090341	0.158411	0.1564	0.09789	0.150393	0.201595	0.197401	0.092223	0.138485	0.237972	0.282536	0.38313	0.336366	0.529386	0.577175	0.633816	0.683599	0.776234	1.384013	5.739646	5.739646	5.739646
3.25	-0.2566	0.145128	0.167771	0.261747	0.28938	0.156385	0.255478	0.282935	0.386633	0.214655	0.315582	0.28938	0.331191	0.408243	0.222687	0.228754	0.34249	0.34022	0.460559	0.502342	0.645445	0.636257	0.759392	0.683068	1.609024	5.739646	5.739646	5.739646
3.35	-0.0441	0.411948	0.493115	0.495605	0.400548	0.470981	0.473416	0.483217	0.515751	0.468551	0.407373	0.461298	0.505627	0.478305	0.384804	0.425801	0.490632	0.531131	0.552014	0.672399	0.666466	0.640196	0.733746	0.614599	1.853991	5.739646	5.739646	5.739646
3.45	0.305986	0.664681	0.754925	0.602624	0.792168	0.773373	0.60787	0.684358	0.713158	0.549141	0.687201	0.637225	0.602624	0.631823	0.477382	0.618445	0.486658	0.486658	0.475077	0.642656	0.539271	0.757976	0.605243	0.733823	2.241307	5.739646	5.739646	5.739646
3.55	0.55521	0.941563	1.009719	0.959053	0.980454	0.871271	0.852063	0.884284	0.820847	0.83011	0.871271	0.868044	0.811669	0.811669	0.764095	0.710391	0.772844	0.874508	0.858425	0.83011	0.718681	0.72146	0.823925	0.73547	3.039749	5.739646	5.739646	5.739646
3.65	0.533403	0.989743	1.072223	1.075968	1.149927	1.09109	1.035523	0.972678	1.042756	0.910366	1.017666	0.936131	0.907191	0.959233	1.014133	1.035523	0.989743	1.061071	0.891468	0.793471	0.807716	0.845731	0.727786	0.857724	4.519719	5.739646	5.739646	5.739646
3.75	0.760512	1.033051	1.270209	1.546938	1.223817	1.133258	1.183432	1.10725	1.118314	1.060672	1.092685	1.07833	0.95449	1.036462	1.067698	1.085482	1.163836	1.029651	1.039885	1.07833	1.125758	1.10725	0.983232	1.13703	5.739646	5.739646	5.739646	5.739646
3.85	0.868745	1.40306	1.344031	1.450962	1.384523	1.522132	1.370843	1.284132	1.296669	1.284132	1.344031	1.243443	1.134022	1.352889	1.181598	1.243443	1.2355	1.379942	1.384523	1.227619	1.300883	1.344031	1.17413	1.522132	5.739646	5.739646	5.739646	5.739646
3.95	0.892594	1.556971	1.399083	1.599318	1.463049	1.399083	1.583227	1.453656	1.385893	1.372875	1.463049	1.439731	1.385893	1.351547	1.506441	1.372875	1.239852	1.593925	1.439731	1.224867	1.364289	1.516344	1.306168	1.769874	5.739646	5.739646	5.739646	5.739646

Figure 7: Acceptor-Orbital-Hydrogen Angles for π -hydrogen interactions represented as a heatmap.

For π -hydrogen interactions, the donor and acceptor heatmaps are very different. The acceptor-orbital-hydrogen distances are similar to the π - π interactions, with the density all being taken up at around 2-2.1 angstroms and at very close to 180^0. With the hydrogens, however, the distance is maintained at about 2 angstroms with a large amount of deviation in the possible angles. This is consistent with what we know about possible hydrogen bonding in these systems[27]. These types of interactions are T-stacked cation-π, salt bridges, and hydrogen bonds.

	-0.975	-0.925	-0.875	-0.825	-0.775	-0.725	-0.675	-0.625	-0.575	-0.525	-0.475	-0.425	-0.375	-0.325
0.05	6.01657	6.01657	6.01657	6.01657	6.01657	6.01657	6.01657	6.01657	6.01657	6.01657	6.01657	6.01657	6.01657	6.01657
0.15	6.01657	6.01657	6.01657	6.01657	6.01657	6.01657	6.01657	6.01657	6.01657	6.01657	6.01657	6.01657	6.01657	6.01657
0.25	6.01657	6.01657	6.01657	6.01657	6.01657	6.01657	6.01657	6.01657	6.01657	6.01657	6.01657	6.01657	6.01657	6.01657
0.35	6.01657	6.01657	6.01657	6.01657	6.01657	6.01657	6.01657	6.01657	6.01657	6.01657	6.01657	6.01657	6.01657	6.01657
0.45	6.01657	6.01657	6.01657	6.01657	6.01657	6.01657	6.01657	6.01657	6.01657	2.714352	6.01657	6.01657	6.01657	6.01657
0.55	6.01657	3.097683	3.097683	6.01657	6.01657	3.097683	6.01657	6.01657	6.01657	6.01657	6.01657	6.01657	6.01657	6.01657
0.65	2.36246	3.410598	3.410598	2.755065	6.01657	3.410598	3.410598	3.410598	6.01657	2.755065	6.01657	2.755065	6.01657	6.01657
0.75	1.200419	1.126889	2.143057	2.143057	1.964205	3.028646	3.67263	3.67263	2.640121	3.67263	6.01657	3.028646	6.01657	6.01657
0.85	0.306934	0.840107	0.799511	0.926606	1.375078	1.613613	1.613613	2.058628	2.880775	3.89603	3.89603	2.880775	6.01657	6.01657
0.95	-1.58317	-1.08243	-0.76549	-0.41877	-0.03275	0.291763	0.944153	1.668314	1.926806	2.818293	6.01657	3.471419	4.089031	3.471419
1.05	-3.13742	-2.61507	-2.29786	-2.01065	-1.59905	-1.08381	-0.51085	0.087071	0.703833	1.724621	2.341332	3.28079	6.01657	4.257462
1.15	-3.84929	-3.34106	-2.92083	-2.61887	-2.23463	-1.84125	-1.3384	-0.8698	-0.2563	0.616527	1.094714	2.403814	2.1229	3.449877
1.25	-3.96754	-3.42909	-2.91384	-2.40251	-2.08451	-1.65579	-1.20203	-0.72349	-0.23067	0.409445	1.004283	1.784785	2.285973	3.128423
1.35	-3.72911	-3.28664	-2.85123	-2.34418	-1.92651	-1.47635	-1.08962	-0.58568	-0.12199	0.391582	0.972313	1.798738	2.095833	2.612678
1.45	-3.3301	-3.03198	-2.66992	-2.3342	-1.88938	-1.40085	-1.00473	-0.57926	-0.07506	0.47295	0.782579	1.212559	1.857371	1.983077
1.55	-2.86241	-2.63807	-2.3365	-2.0727	-1.74296	-1.38862	-0.98814	-0.52228	-0.12389	0.377623	0.875698	1.1744	1.657157	1.715222
1.65	-2.34739	-2.15316	-1.93544	-1.69145	-1.50869	-1.13653	-0.85518	-0.43191	-0.12401	0.370144	0.542191	0.974485	1.183536	1.247544
1.75	-1.82327	-1.64144	-1.40773	-1.1996	-1.12163	-0.89765	-0.58064	-0.3537	0.010307	0.311469	0.700994	0.818163	1.169751	1.45031
1.85	-1.2393	-1.06929	-0.86433	-0.82043	-0.54028	-0.36788	-0.27195	0.012243	0.229057	0.308053	0.557687	0.786376	0.88198	1.45783
1.95	-0.74957	-0.49284	-0.46761	-0.2363	-0.11976	0.065145	0.113443	0.269678	0.487587	0.662502	0.562241	0.766605	0.874637	1.176772
2.05	-0.32202	-0.09541	-0.01695	0.232386	0.17583	0.259915	0.632449	0.522926	0.615236	0.768635	0.802355	0.84438	1.02414	1.104331
2.15	0.28185	0.370928	0.710041	0.628195	0.582121	0.756596	0.733048	0.750657	0.774628	0.811699	0.850197	1.005578	1.085065	1.162442
2.25	0.739855	0.84109	0.713462	1.183605	0.877248	0.908412	1.001221	1.103534	1.029401	0.940579	0.852998	0.987424	0.973814	1.05107
2.35	1.130136	1.122918	1.137406	1.0738	1.152108	1.130136	1.14473	1.189836	1.221079	1.066976	1.182176	1.046779	1.152108	1.174574
2.45	1.327778	1.439355	1.534544	1.335932	1.448476	1.177268	1.319689	1.090334	1.257234	1.344154	0.964141	1.058812	0.975496	1.136204
2.55	1.500527	1.57448	1.284945	1.375078	1.242753	1.440022	1.306728	1.101698	1.107696	1.107696	1.066448	0.978103	1.23589	1.010325
2.65	1.689567	1.516132	1.420307	1.541594	1.45124	1.325915	1.214561	1.137061	1.125654	1.196136	1.233333	1.049249	1.054517	1.196136
2.75	1.714047	1.685989	1.548347	1.399292	1.385539	1.294227	1.257509	1.008771	1.061686	1.122797	1.149404	1.076617	1.091775	1.122797
2.85	1.476939	1.587319	1.579569	1.483931	1.304575	1.236573	1.113043	1.20423	1.061153	1.038428	0.994471	0.981652	1.038428	1.02948
2.95	1.510933	1.490951	1.378783	1.372906	1.176658	1.089102	1.220969	1.033327	1.071607	1.148172	0.984466	1.033327	1.115932	1.029163
3.05	1.433076	1.365542	1.524461	1.107905	1.133465	1.091221	1.038843	0.843364	1.091221	0.893001	0.985345	1.129159	1.054669	1.120603
3.15	1.376469	1.301946	1.201804	1.155259	1.072	0.936983	1.038336	0.981151	0.967352	1.075812	1.034665	1.056898	1.027362	1.184627
3.25	1.229739	1.229739	1.14168	1.082181	0.985885	0.92237	0.901059	0.892064	0.966403	0.937874	1.149372	1.137856	1.134046	1.330927
3.35	1.244853	1.171636	0.928711	1.039554	0.93163	0.955291	1.113883	0.925801	0.970369	1.076028	1.042815	1.117396	1.175359	1.289829
3.45	1.085029	1.05965	1.0349	0.981355	0.925041	1.001839	1.101224	1.154884	1.001839	1.267835	1.131052	1.107776	1.25633	1.382131
3.55	1.113257	1.073482	1.085553	1.052702	1.061555	1.094703	1.167779	1.154681	1.228944	1.01522	1.157939	1.347877	1.316791	1.235977
3.65	1.143739	1.131636	1.162172	1.15599	1.180951	1.341698	1.184115	1.18729	1.239502	1.159076	1.162172	1.4029	1.30885	1.480804
3.75	1.16151	1.362406	1.334101	1.413954	1.303187	1.303187	1.327148	1.286417	1.341103	1.289749	1.472302	1.380513	1.436886	1.476315
3.85	1.488967	1.335256	1.379286	1.379286	1.477441	1.396751	1.389729	1.458522	1.586289	1.48511	1.428978	1.393234	1.661447	1.556912
3.95	1.599356	1.543552	1.571065	1.749454	1.563127	1.516776	1.465284	1.505517	1.535829	1.535829	1.528164	1.531989	1.591191	1.658459

	-0.275	-0.225	-0.175	-0.125	-0.075	-0.025	0.025	0.075	0.125	0.175	0.225	0.275	0.325	0.375
0.05	6.01657	6.01657	6.01657	6.01657	6.01657	6.01657	6.01657	6.01657	6.01657	6.01657	6.01657	6.01657	6.01657	6.01657
0.15	6.01657	6.01657	6.01657	6.01657	6.01657	6.01657	6.01657	6.01657	6.01657	6.01657	6.01657	6.01657	6.01657	6.01657
0.25	6.01657	6.01657	6.01657	6.01657	6.01657	6.01657	6.01657	6.01657	6.01657	6.01657	6.01657	6.01657	1.564551	6.01657
0.35	6.01657	6.01657	6.01657	6.01657	6.01657	6.01657	6.01657	6.01657	6.01657	6.01657	6.01657	6.01657	6.01657	6.01657
0.45	6.01657	6.01657	6.01657	6.01657	6.01657	6.01657	6.01657	6.01657	6.01657	6.01657	6.01657	6.01657	6.01657	2.714352
0.55	6.01657	6.01657	6.01657	6.01657	6.01657	6.01657	6.01657	6.01657	6.01657	6.01657	6.01657	6.01657	6.01657	6.01657
0.65	6.01657	6.01657	3.410598	3.410598	6.01657	6.01657	6.01657	6.01657	6.01657	6.01657	6.01657	6.01657	6.01657	6.01657
0.75	6.01657	3.67263	6.01657	6.01657	3.67263	6.01657	6.01657	6.01657	3.67263	6.01657	6.01657	6.01657	6.01657	6.01657
0.85	6.01657	6.01657	3.89603	6.01657	3.89603	6.01657	6.01657	6.01657	6.01657	3.89603	6.01657	6.01657	6.01657	6.01657
0.95	6.01657	4.089031	6.01657	4.089031	6.01657	4.089031	6.01657	6.01657	6.01657	6.01657	6.01657	6.01657	6.01657	6.01657
1.05	4.257462	6.01657	4.257462	6.01657	6.01657	4.257462	6.01657	6.01657	6.01657	6.01657	6.01657	6.01657	6.01657	6.01657
1.15	4.405621	4.405621	4.405621	2.970249	4.405621	6.01657	6.01657	4.405621	6.01657	6.01657	6.01657	4.405621	4.405621	6.01657
1.25	3.9645	3.9645	4.536785	3.602801	6.01657	3.9645	6.01657	4.536785	6.01657	4.536785	6.01657	4.536785	6.01657	6.01657
1.35	2.714352	3.480307	4.09727	3.741944	6.01657	4.65352	6.01657	4.09727	4.65352	6.01657	6.01657	6.01657	6.01657	4.65352
1.45	2.964152	2.658667	3.236208	3.869186	3.611136	3.869186	3.869186	6.01657	6.01657	6.01657	6.01657	6.01657	6.01657	6.01657
1.55	2.113935	2.31013	2.703387	2.55445	3.216716	3.731727	4.851535	3.986042	4.327938	6.01657	4.851535	6.01657	6.01657	6.01657
1.65	2.03573	2.281725	2.192831	2.329322	3.097683	3.208739	3.333688	3.476509	4.93587	4.428666	6.01657	4.428666	4.428666	6.01657
1.75	1.815417	1.584927	2.307788	2.151082	2.493697	3.019003	3.208644	3.442868	4.193338	4.193338	4.521147	3.94687	6.01657	6.01657
1.85	1.634826	1.736652	2.334856	1.977937	3.041913	3.214073	3.041913	3.124293	3.545084	3.848331	5.081035	6.01657	5.081035	5.081035
1.95	1.499157	1.530063	1.906774	1.929914	2.196613	3.223442	3.410598	3.779813	3.312648	3.312648	4.133253	4.684827	3.940998	5.14368
2.05	1.20169	1.48328	1.6447	1.6289	1.982493	2.424413	2.851062	3.23565	3.090394	3.23565	4.217402	4.027905	4.757453	5.200695
2.15	1.337865	1.296137	1.403916	1.633131	1.911922	2.418066	2.666049	2.482872	2.841382	3.180307	3.493592	3.324731	3.816465	3.816465
2.25	1.111259	1.208933	1.406842	1.722869	1.893946	2.361017	2.445682	2.604974	2.714352	3.14015	3.67263	3.200939	4.031236	3.896899
2.35	1.237074	1.295161	1.269852	1.48255	1.852375	1.946132	2.501147	2.343098	2.501147	2.760303	3.059865	3.569295	3.569295	4.945185
2.45	1.2802	1.212825	1.287974	1.466972	1.534544	1.673609	2.061258	1.949378	2.499971	2.474085	3.189648	2.877858	3.731166	3.565945
2.55	1.15702	1.11373	1.037992	1.431669	1.491656	1.57448	1.53682	1.848734	2.139674	2.526514	2.812318	2.688038	3.075739	4.113642
2.65	1.114375	1.131341	1.208382	1.412721	1.491301	1.541594	1.567723	1.783558	2.003081	2.059177	2.073707	2.65188	2.99018	3.336583
2.75	1.128062	1.004096	1.187886	1.471044	1.263536	1.399292	1.614801	1.614801	1.834886	1.752727	2.254217	2.393751	2.600986	2.697617
2.85	1.052001	1.079712	1.25873	1.310456	1.32831	1.442697	1.476939	1.409589	1.643319	1.729185	1.813321	2.05484	2.323935	2.444703
2.95	1.029163	0.996459	1.231089	1.241312	1.464915	1.445823	1.349735	1.609854	1.695083	1.671109	1.788259	1.824416	2.186462	2.313139
3.05	1.133465	1.091221	1.214315	1.360115	1.365542	1.370999	1.438915	1.450697	1.668293	1.69064	1.576844	1.72903	1.976427	2.038459
3.15	1.176147	1.193178	1.306745	1.351006	1.392064	1.397317	1.614066	1.491116	1.52644	1.581882	1.556861	1.614066	1.832523	1.89118
3.25	1.317134	1.250882	1.383198	1.524429	1.428095	1.55872	1.61862	1.60635	1.631044	1.612466	1.570416	1.800951	1.637313	1.815877
3.35	1.248859	1.359085	1.540406	1.482984	1.545793	1.768778	1.624434	1.716175	1.672313	1.755366	1.735579	1.742131	1.782373	1.607054
3.45	1.315229	1.395189	1.501404	1.724248	1.625758	1.761462	1.767802	1.724248	1.87511	2.00335	1.854093	1.767802	1.840323	1.903838
3.55	1.404715	1.487415	1.567693	1.582554	1.676678	1.824109	1.704476	1.792832	1.836899	1.917275	1.903422	1.896567	2.067507	2.160892
3.65	1.584028	1.607951	1.65251	1.726028	1.853728	2.074531	1.911063	1.951199	1.985921	2.138291	2.051632	2.130095	2.188928	2.074531
3.75	1.509017	1.59126	1.632712	1.675956	1.86453	1.984053	2.031958	1.870477	2.112203	2.150921	2.097126	2.351323	2.1912	2.150921
3.85	1.652306	1.675317	1.679984	1.892894	1.809649	1.946383	2.022467	2.002897	2.126514	2.111999	2.15619	2.186774	2.171365	2.2509
3.95	1.754229	1.865486	1.763847	1.972664	1.914826	2.184052	1.914826	2.140863	2.106244	2.276465	2.221526	2.42399	2.26843	2.414683

Figure 8: Donor-Hydrogen-Orbital Angles for lone pair-hydrogen interactions represented as a heatmap.

	-0.975	-0.925	-0.875	-0.825	-0.775	-0.725	-0.675	-0.625	-0.575	-0.525	-0.475	-0.425	-0.375	-0.325
0.05	6.091197	6.091197	6.091197	6.091197	6.091197	6.091197	6.091197	6.091197	6.091197	6.091197	6.091197	6.091197	6.091197	6.091197
0.15	6.091197	6.091197	6.091197	6.091197	6.091197	6.091197	6.091197	6.091197	6.091197	6.091197	6.091197	6.091197	6.091197	6.091197
0.25	1.639178	6.091197	6.091197	6.091197	6.091197	6.091197	6.091197	6.091197	6.091197	6.091197	6.091197	6.091197	6.091197	6.091197
0.35	6.091197	6.091197	6.091197	6.091197	6.091197	6.091197	6.091197	6.091197	6.091197	6.091197	6.091197	6.091197	6.091197	6.091197
0.45	6.091197	6.091197	6.091197	6.091197	6.091197	6.091197	6.091197	6.091197	6.091197	6.091197	6.091197	6.091197	6.091197	6.091197
0.55	6.091197	3.17231	3.17231	6.091197	6.091197	6.091197	6.091197	6.091197	6.091197	6.091197	6.091197	6.091197	6.091197	3.17231
0.65	2.155896	2.155896	3.485225	3.485225	2.829692	6.091197	3.485225	3.485225	6.091197	6.091197	3.485225	3.485225	6.091197	6.091197
0.75	0.951875	2.217683	3.103273	2.217683	2.038832	2.714747	2.714747	2.435644	3.103273	2.714747	6.091197	3.747256	6.091197	3.747256
0.85	0.224415	0.660077	0.797579	1.001232	1.886194	1.886194	3.339367	3.970656	2.678645	3.970656	2.678645	3.339367	3.339367	6.091197
0.95	-1.74095	-1.05516	-0.39911	0.01454	0.705394	0.84986	1.367542	1.601565	1.907646	1.74294	2.350958	2.501159	2.501159	4.163658
1.05	-3.25093	-2.69799	-2.0529	-1.62141	-0.91791	-0.62763	-0.04544	0.440013	0.93179	2.018977	1.799247	1.675695	2.300995	2.695228
1.15	-3.5348	-3.21425	-2.96939	-2.69102	-2.34298	-1.96553	-1.59087	-0.85478	-0.37994	0.018575	0.621256	1.169341	1.600693	1.694882
1.25	-3.14267	-2.92558	-2.75937	-2.63972	-2.50692	-2.33682	-2.19694	-1.94947	-1.62744	-1.27289	-0.8515	-0.23189	0.148649	0.645943
1.35	-2.65432	-2.53134	-2.39228	-2.2997	-2.26067	-2.09702	-1.96503	-1.92361	-1.78682	-1.63732	-1.54486	-1.25547	-1.04552	-0.69774
1.45	-1.95657	-2.0107	-1.95818	-1.9844	-1.88038	-1.89848	-1.86724	-1.77689	-1.64616	-1.54682	-1.37354	-1.22764	-1.16914	-1.08989
1.55	-1.14007	-1.24168	-1.2964	-1.42917	-1.41714	-1.45513	-1.53136	-1.46556	-1.5029	-1.48836	-1.32467	-1.27292	-1.01978	-0.94116
1.65	-0.68535	-0.72025	-0.68978	-0.75189	-0.83361	-0.91269	-0.90024	-1.04346	-0.98926	-1.14095	-1.05118	-1.071	-1.00389	-0.925
1.75	-0.09413	-0.17948	-0.34499	-0.23052	-0.30539	-0.17948	-0.42234	-0.46505	-0.48935	-0.59409	-0.57003	-0.63849	-0.58323	-0.65896
1.85	0.208652	0.191531	0.187297	0.261845	0.195784	0.141862	0.187297	0.08687	-0.062	-0.00089	-0.08157	-0.14416	-0.18284	-0.18284
1.95	0.362265	0.482476	0.595978	0.467265	0.427796	0.584596	0.704704	0.524211	0.305036	0.452281	0.418167	0.215075	0.172959	0.176715
2.05	0.391983	0.494161	0.623829	0.75448	0.71288	0.797886	0.849916	0.823562	0.785291	0.791568	0.571949	0.742416	0.64536	0.623829
2.15	0.617502	0.944643	0.873815	0.825283	0.81351	0.951337	0.978574	0.867619	0.751121	0.971695	1.042711	0.837197	0.958077	1.006573
2.25	0.793311	0.793311	0.939676	1.002214	0.933632	0.983039	1.111199	1.055223	1.096908	1.041705	1.104028	1.22543	1.118421	1.283559
2.35	0.967295	0.984582	0.955934	1.026121	1.162216	1.121405	1.155297	1.169182	1.279962	1.044465	1.204763	1.249201	1.3117	1.256803
2.45	0.825989	0.95748	0.962699	1.02754	1.204147	1.021973	1.251895	0.95748	1.152232	1.251895	1.354826	1.410559	1.33186	1.316837
2.55	0.765347	0.891978	0.905569	0.914734	1.042216	1.079508	1.123904	1.095927	0.942744	1.237987	1.219086	1.270304	1.388723	1.418749
2.65	0.941839	0.733098	0.772984	0.991215	1.005116	1.082702	1.145117	1.295406	1.166818	1.092838	1.258665	1.301663	1.295406	1.599174
2.75	0.78225	0.813951	0.865352	0.835657	0.968444	1.02427	0.981053	0.989548	1.051126	1.251366	1.262513	1.268133	1.268133	1.568241
2.85	0.720453	0.705394	0.729598	0.917735	0.940163	0.899422	1.002554	1.014698	1.131193	1.073407	1.073407	1.202302	1.252682	1.433421
2.95	0.666482	0.60659	0.810557	1.012508	0.904739	1.107954	0.964376	1.129039	1.020123	1.177054	1.075116	1.22749	1.251285	1.441689
3.05	0.567555	0.611722	0.762805	0.695981	0.90516	0.930988	0.984737	0.977858	1.199499	1.074967	1.090191	1.261259	1.293631	1.2477
3.15	0.645545	0.546861	0.620508	0.717196	0.846537	0.93179	1.011609	0.985367	1.041979	1.0733	1.083962	1.165831	1.209424	1.391039
3.25	0.613298	0.578279	0.664923	0.742946	0.920034	0.843282	0.996996	0.934381	1.093848	0.981729	1.08038	1.227868	1.325509	1.351485
3.35	0.612668	0.671554	0.628569	0.817679	0.824967	0.808043	1.003338	0.963346	1.048039	0.949442	1.137237	1.13391	1.181519	1.470226
3.45	0.617512	0.695184	0.83356	0.824327	0.975326	0.905664	0.951562	0.994207	1.019019	1.097378	1.140562	1.222643	1.185695	1.410286
3.55	0.716373	0.734059	0.799606	0.881171	0.969984	0.969984	1.133222	1.067461	1.078591	1.092681	1.127329	1.194146	1.300073	1.368722
3.65	0.878432	0.876268	0.882775	1.00287	1.017703	1.153544	1.06618	1.173717	1.098054	1.206263	1.33102	1.320851	1.438839	1.465762
3.75	0.885633	0.985574	1.146353	1.10689	1.096624	1.159861	1.099181	1.165316	1.215796	1.334783	1.331548	1.419249	1.499981	1.596189
3.85	1.01084	1.091848	1.18528	1.154011	1.270897	1.256583	1.24247	1.239671	1.294236	1.248091	1.24247	1.563593	1.529408	1.639845
3.95	1.133184	1.18461	1.296148	1.359948	1.215229	1.282199	1.310295	1.348041	1.345086	1.359948	1.565325	1.565325	1.768496	1.768496

	-0.275	-0.225	-0.175	-0.125	-0.075	-0.025	0.025	0.075	0.125	0.175	0.225	0.275	0.325	0.375
0.05	6.091197	6.091197	6.091197	6.091197	6.091197	6.091197	6.091197	6.091197	6.091197	6.091197	6.091197	6.091197	6.091197	6.091197
0.15	6.091197	6.091197	6.091197	6.091197	6.091197	6.091197	6.091197	6.091197	6.091197	6.091197	6.091197	6.091197	6.091197	6.091197
0.25	6.091197	6.091197	6.091197	6.091197	6.091197	6.091197	6.091197	6.091197	6.091197	6.091197	6.091197	6.091197	6.091197	6.091197
0.35	6.091197	6.091197	6.091197	6.091197	6.091197	6.091197	6.091197	6.091197	6.091197	6.091197	6.091197	6.091197	6.091197	6.091197
0.45	6.091197	6.091197	6.091197	6.091197	6.091197	6.091197	6.091197	6.091197	2.788979	6.091197	6.091197	6.091197	6.091197	6.091197
0.55	6.091197	6.091197	6.091197	6.091197	6.091197	6.091197	6.091197	6.091197	6.091197	6.091197	6.091197	6.091197	6.091197	6.091197
0.65	6.091197	6.091197	6.091197	6.091197	6.091197	6.091197	6.091197	6.091197	6.091197	3.485225	6.091197	6.091197	6.091197	6.091197
0.75	2.714747	6.091197	6.091197	6.091197	6.091197	6.091197	6.091197	6.091197	3.747256	6.091197	6.091197	6.091197	6.091197	6.091197
0.85	6.091197	6.091197	6.091197	6.091197	2.955402	3.970656	6.091197	6.091197	6.091197	6.091197	6.091197	6.091197	6.091197	6.091197
0.95	2.892919	6.091197	6.091197	3.167082	6.091197	3.546046	4.163658	6.091197	6.091197	6.091197	6.091197	6.091197	6.091197	6.091197
1.05	3.084078	6.091197	4.332088	4.332088	3.355417	3.728974	4.332088	4.332088	6.091197	6.091197	6.091197	6.091197	6.091197	6.091197
1.15	1.694882	2.119052	2.870508	3.044875	2.870508	4.480248	4.480248	3.892293	6.091197	6.091197	6.091197	6.091197	6.091197	6.091197
1.25	1.056584	1.224462	2.07893	3.030051	2.282403	3.412369	4.039126	4.611412	4.039126	4.611412	6.091197	6.091197	6.091197	6.091197
1.35	-0.30099	0.293273	0.957345	1.679288	1.752463	2.51054	2.687305	3.554934	3.816571	6.091197	6.091197	6.091197	6.091197	6.091197
1.45	-1.01835	-0.70943	0.011035	0.335135	1.373452	1.719914	2.733293	3.165583	3.48083	4.832506	4.832506	6.091197	6.091197	6.091197
1.55	-0.89737	-0.68003	-0.47596	-0.14869	0.149873	0.730941	1.302656	1.789849	2.953082	4.060669	4.926162	4.926162	6.091197	6.091197
1.65	-0.85444	-0.53986	-0.09847	0.042779	0.144308	0.479736	0.728991	1.373018	2.185826	3.717808	5.010497	6.091197	6.091197	6.091197
1.75	-0.51073	-0.35602	-0.22428	0.072226	0.124056	0.575844	0.742167	1.178632	1.745485	2.568324	3.28327	5.086671	4.267965	6.091197
1.85	-0.03191	-0.06529	0.239336	0.337479	0.372457	0.568454	1.114342	1.424902	1.948967	2.841706	3.496793	4.360298	4.680905	6.091197
1.95	0.353245	0.584596	0.679501	0.704704	0.704704	0.990861	1.218864	1.343718	2.004541	2.511979	3.387275	3.85444	5.218306	6.091197
2.05	0.634537	0.730497	0.904802	0.985554	0.926186	1.46728	1.557907	1.59941	2.253864	2.49904	3.310277	4.102532	5.275321	6.091197
2.15	1.119159	1.184825	1.467226	1.350534	1.501566	1.736799	2.005643	2.025109	2.461799	2.916009	3.324541	4.370886	6.091197	4.60061
2.25	1.147845	1.37294	1.557217	1.702127	2.020799	1.918939	2.463077	2.114404	2.751174	2.828268	3.340292	4.261087	5.374864	5.374864
2.35	1.536777	1.506936	1.841286	1.677485	1.827688	2.393652	2.417725	2.467683	3.039372	3.185674	3.490269	4.514521	5.019811	5.019811
2.45	1.589394	1.670962	1.759778	2.039241	1.97964	2.261876	2.406062	2.685414	3.164851	3.317957	3.719774	4.003821	4.402056	5.073819
2.55	1.548773	1.658747	1.838694	1.975168	2.133255	1.874106	2.148947	2.601141	2.529214	2.886945	3.288664	4.188269	4.854211	5.124159
2.65	1.487348	1.869195	1.794553	1.937931	2.050797	2.077707	2.105362	2.224329	2.256421	2.726507	3.579911	4.528099	4.253346	5.528889
2.75	1.647379	1.639178	1.680876	1.655649	1.95327	2.220926	2.084525	2.123528	2.431627	2.722762	2.999028	3.42984	3.71058	5.560082
2.85	1.601565	1.717946	1.717946	1.734539	1.907646	2.001433	1.948248	1.979818	2.141961	2.398562	2.366575	2.919924	3.77519	4.263416
2.95	1.459322	1.45341	1.605949	1.761655	1.684481	1.737871	1.737871	1.965643	2.11384	2.160555	2.184757	2.402885	3.267764	3.559773
3.05	1.346718	1.531267	1.605488	1.685661	1.788123	1.765266	1.757762	1.893694	2.123809	2.081589	2.051054	2.498722	2.707248	3.164712
3.15	1.542867	1.589153	1.589153	1.662863	1.813967	1.763368	1.791968	1.983218	1.974475	1.770442	2.265435	2.402512	2.389245	2.660058
3.25	1.448115	1.555091	1.616054	1.656878	1.73742	1.770212	1.680977	1.944574	1.890504	1.868197	1.968673	2.374167	2.755574	2.703382
3.35	1.542506	1.557611	1.62042	1.759277	1.927892	1.740828	1.74694	1.863867	1.942695	1.716748	2.086808	2.04465	2.535759	2.761884
3.45	1.561519	1.547216	1.717075	1.745527	1.956842	1.728359	1.678555	1.762995	1.921811	2.054118	1.963998	2.188496	2.606418	2.769148
3.55	1.504726	1.637415	1.632534	1.703135	1.837131	1.68245	1.837131	1.756803	1.971194	2.226659	2.191987	2.102407	2.36857	2.689398
3.65	1.568305	1.62153	1.874129	1.795221	1.687432	1.880011	1.972662	1.845227	2.196592	2.188529	2.335326	2.485588	2.382948	2.529787
3.75	1.712052	1.712052	1.909941	1.800932	1.785559	1.969251	1.963159	2.01937	2.135025	2.20988	2.369678	2.548988	2.714747	2.538137
3.85	1.731493	1.842559	1.7879	1.956013	1.990941	2.077523	2.02101	2.144305	2.20114	2.238376	2.611013	2.845798	2.804862	3.028686
3.95	1.824081	1.972734	2.041354	2.023753	2.127858	2.096098	2.089865	2.2735	2.280994	2.335085	2.576368	2.827845	3.164609	2.937628

Figure 9: Acceptor-Orbital-Hydrogen Angles for lone pair-hydrogen interactions represented as a heatmap.

For lone pair hydrogen interactions we see some variation between the Donor distances and angles and the Acceptor distances and angles, but the variation is not as pronounced as it is between hydrogens and π orbitals. As before the Donor angles are centred very close to 180^0, but in this case the distances are closer, around 1.25-1.35 angstroms. Acceptor angles are slightly broader, as they were in the case of π orbitals, but the distances are close to 1.25 angstroms. These interactions include T-stacked and offset parallel π-π interactions.

As shown in Figure 11, placing explicit orbitals allows for accounting the π -stacking as well as cation- π interactions and other PCI interactions that are not captured in a simple hydrogen bonding statistical potential in *score12* or *talaris2013* or *talaris2014*.

The other advantage that this method has over the hydrogen bonding potential is that placement of an explicit orbital reduces the information needed to score the same interaction. For example, as shown in Figure 11, the previous hydrogen bonding potential requires a distance, two angles and a torsion to score a hydrogen bond. These attempt to capture the correct geometry. Because there is no lone pair explicit, the geometry of the interaction is captured using the atoms that are there. This takes a distance, two angles, and a torsion. By placing the orbitals, it is possible to explicitly capture the geometry of PCIs. The previous hydrogen bonding potential had to attempt to capture this explicitly through the use of a torsion, and because of that also required knowledge of the location of a bonded atom.

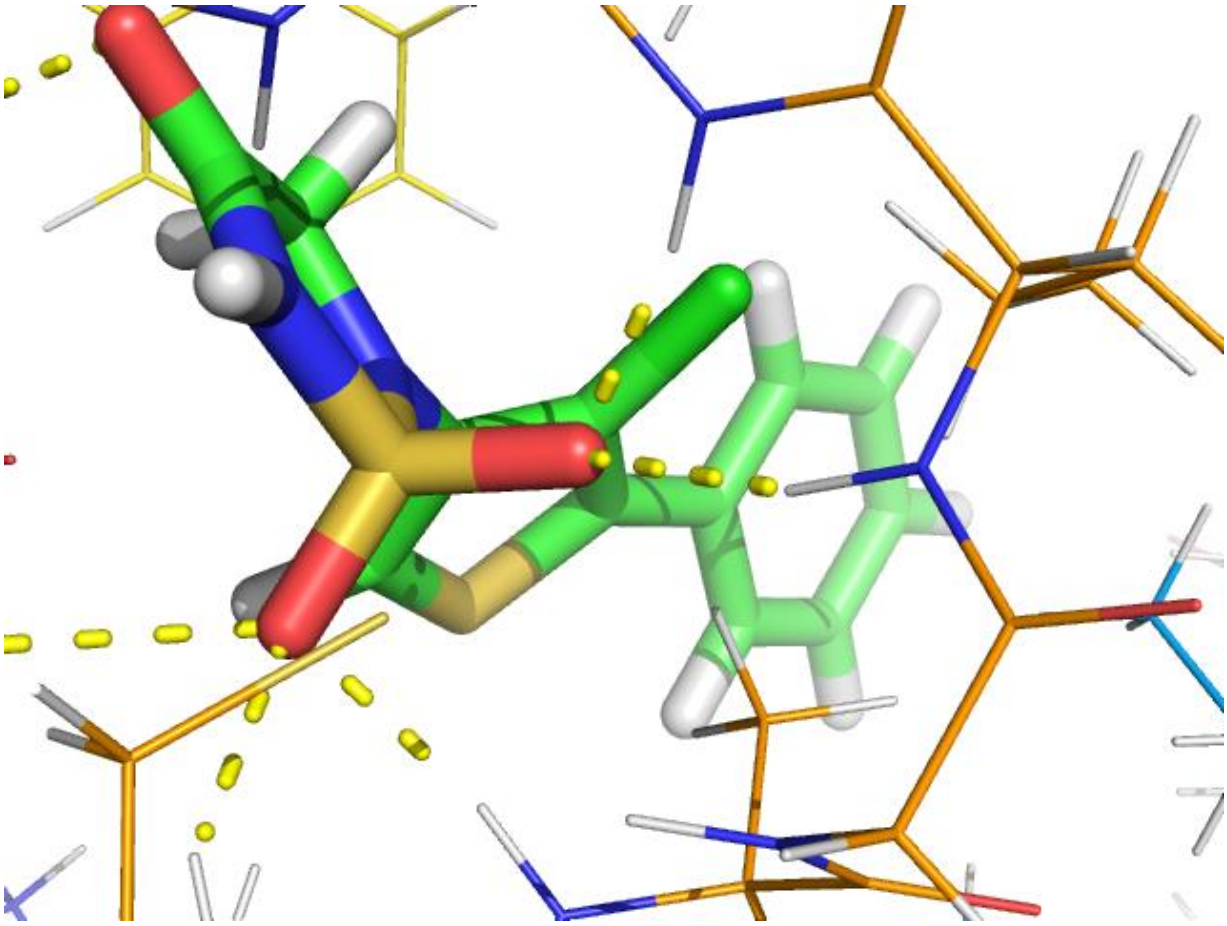


Figure 10: The ligand that is in the binding pocket for model 1023. The picture is zoomed in on a hydrogen bond between the ligand and the inside of the binding pocket.

Figure 11: Flow Chart of AssignOrbitals Procedure

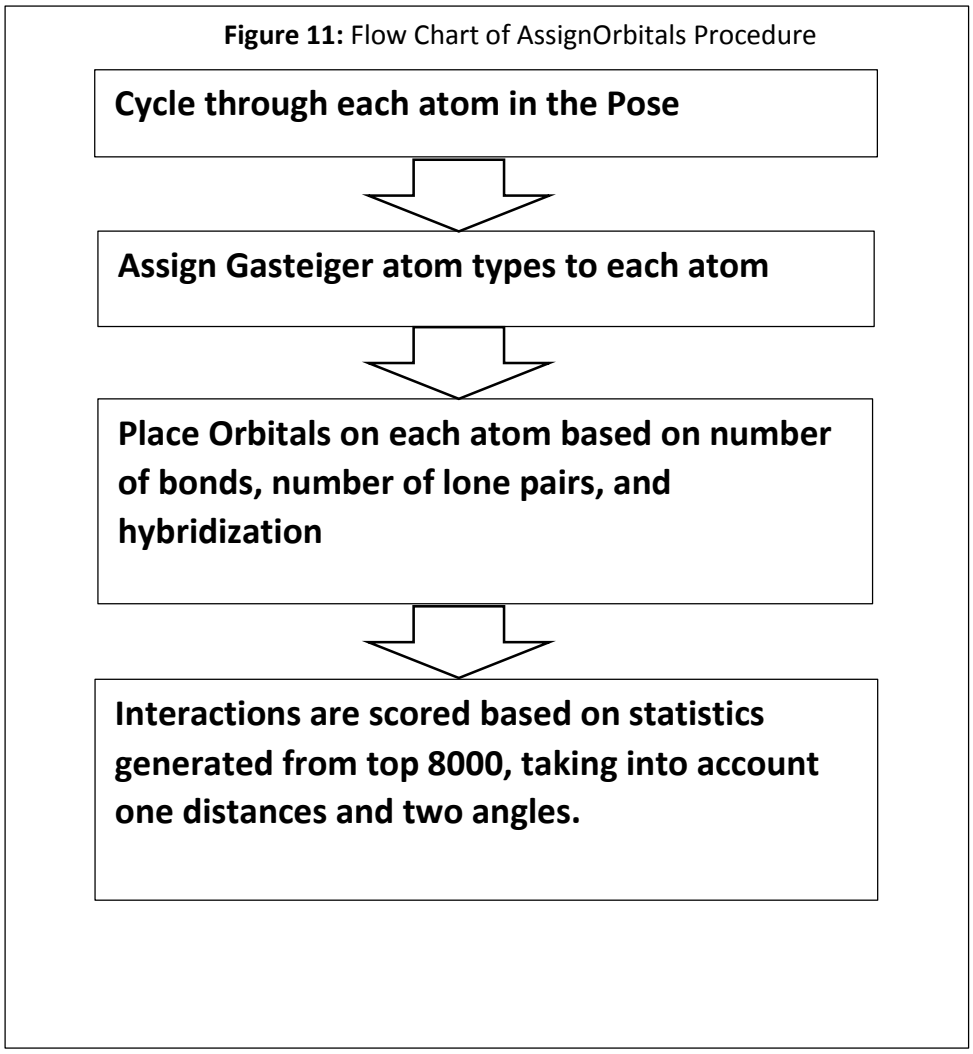
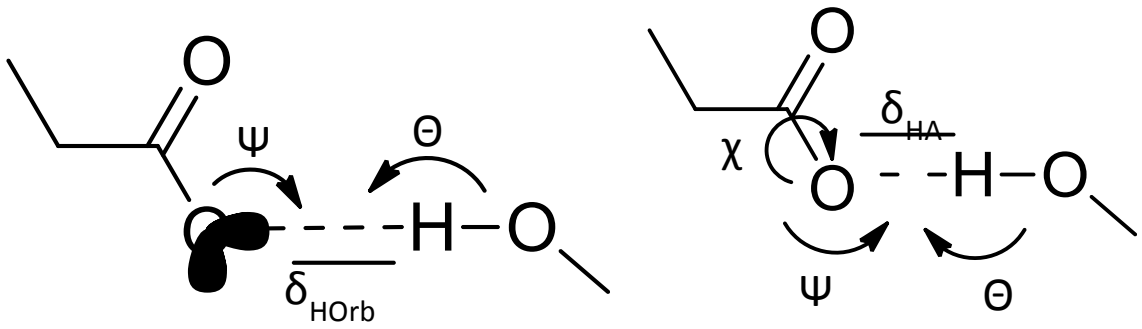


Figure 12: The values that are used to capture Hydrogen Bonding interactions (Modelled on the left). 1) a distance (δ_{HOrb}) between the orbital and hydrogen, 2) the angle Ψ between the acceptor – orbital – hydrogen (AOH), and Θ angle between the donor – hydrogen – orbital (DHO) angle. This is compared to the previous method, using a distance, two angles, and a torsion χ (image on the right).



Benchmarking AssignOrbitals

A benchmark study was carried out to access the *AssignOrbital* program for protein-ligand docking. The benchmark study used a previously outlined set of protein ligand complexes. Each of the protein-ligand complex was docked using *Score12*, *Talaris2013*, *Talaris2014*, and OrbitalScorefunction using the standard docking procedure[28] in Rosetta, generating 5000 models. The score vs RMSD plots were compared[29] for each of the four scoring functions. Results were examined to find which score function could correctly identify the best structure by RMSD to native structure within its top 10 ranking models by score as shown in Table 2. Both of these metrics are used to show how well a score function can properly differentiate between good and bad models. If the score vs RMSD plot is poor, the score function is not good at differentiating between good and bad models. A poor score in the top few models suggests two possibilities. The first is that the mover is not correctly sampling the entire space. The other possibility is that the score function is identifying a particular geometry or interaction as repulsive or neutral when it should be attractive. In either case it is important to correctly identify how the score function performs and what its limitations are in order to make corrections and improvements.

Table 2: **Best RMSD among top 10 models** Best RMSD (in Å) among top 10 scoring models for each of the four score functions. The left hand column is the numerical classification for each of the protein-ligand complexes. Each other column is the highest RMSD to native among the top 10 models by score.

Model	Talaris2014	Orbitals	Talaris2013	Score12
1008	0.40	0.56	0.62	2.37
1042	0.32	0.33	0.36	0.35
1043	0.56	0.48	0.55	0.74
1078	0.53	0.55	0.51	0.49
1079	1.16	0.36	4.80	0.39
1093	0.20	0.19	0.14	0.18
1094	0.31	1.04	0.15	0.76
1097	0.22	1.26	2.41	0.94
1099	0.11	0.28	0.10	0.13
1100	0.28	0.29	0.25	0.27
1110	1.36	1.53	1.67	0.14
1123	0.26	0.29	0.31	0.36
1127	0.52	3.51	0.58	5.80
1144	0.35	0.40	0.36	0.47
1173	0.20	0.39	0.21	0.22
1194	0.16	0.13	0.19	0.11
2023	0.22	0.20	0.25	0.22
2062	0.66	0.54	0.37	0.25
2071	0.26	0.28	0.27	0.21
2087	4.93	3.11	3.79	0.34
2088	0.23	0.52	0.52	0.30
2089	2.56	2.15	3.57	0.43
2092	0.41	0.45	0.50	0.33
2104	0.38	0.24	0.22	0.18
2110	4.52	0.60	4.71	4.39
2129	0.17	0.16	0.21	0.16
2167	0.30	0.48	0.37	0.56
2202	4.55	2.96	3.01	0.80
2261	1.90	1.64	2.21	1.40
2266	0.35	0.33	0.33	4.79
Average	0.95(0.25)	0.84(0.17)	1.12(0.26)	0.93(0.27)

Table 2 shows a comparison of the four score functions in achieving the best docked structure. This shows that on average, *Talaris2014* and *Score12* perform reasonably similarly, with both of them performing about 15% better than *Talaris2013*. The orbital score function performs the best on average, performing about 12% better than *Talaris2014* and 10% better than Score12. Looking at the individual models, some interesting things can be noted. The first model, 1008, both *Talaris* score functions and the orbital score functions give good scoring models that are reasonably close to native (between .4 and .65 RMSD to native) but *Score12* performs much worse than those

three. *Score12* underperforms for models 1127 and 2266. On the other hand it overperforms for models 1110, 2087, and 2202. Clear advantage defined when its best RMSD is 1Å better than the next best score function's best RMSD.

Talaris2014 performs better than or equal to *Talaris2013* in almost all cases. For this reason and because they contain the same scoring terms with slight changes in weighting and the potentials used, they will be considered together. The *Talaris* score functions are the clear losers for 1079, 2110, 2087, and 2202. It is only the clear winner for 1127, but is reasonably better in a few cases.

The orbital score function clearly the best only in 2110, but is also never clearly outperformed by all of the other score functions. This, combined with the fact that the orbital score function performs better on average, suggests that the orbital score function is consistently a strong option for ligand benchmarking. It performs poorly in a few cases, but in those cases one of the three other score functions performed equally as bad or worse.

As can be seen from table two, *Score12* outperformed the other three score functions on average. This suggests that it is still better at discriminating good models from bad models than the other score functions. It did not outperform the orbital score function by a large portion, however, and both the orbital score function and *Score12* outperform both *Talaris* score functions by a significant margin.

Table 3: **Funnel Analysis Results.** The funnel analysis[20] results for the model with the best RMSD among the top 10 models. Numbers between 0 and -1 suggest good discrimination between models with low RMSD and those with high RMSD, with numbers closer to -1 suggesting better discrimination. Numbers between 0 and 1 suggest that atoms with large RMSD to native score better than those with lower RMSD to native.

Model	Talaris2014	Orbitals	Talaris2013	Score12
1008	-0.29	-0.24	-0.41	-0.17
1042	-0.20	-0.18	-0.16	-0.10
1043	-0.05	-0.05	-0.02	-0.05
1078	-0.18	-0.22	-0.32	-0.31
1079	-0.18	-0.26	-0.16	-0.29
1093	-0.15	-0.05	-0.14	-0.10
1094	-0.17	-0.10	-0.18	-0.16
1097	-0.14	-0.52	-0.26	-0.37
1099	-0.26	-0.40	-0.34	-0.38
1100	-0.39	-0.42	-0.37	-0.42
1110	-0.08	-0.04	-0.02	-0.32
1123	-0.14	-0.21	-0.17	-0.20
1127	-0.16	-0.06	-0.15	-0.17
1144	-0.12	-0.21	-0.14	-0.11
1173	-0.45	-0.39	-0.59	-0.46
1194	0.17	-0.19	0.11	-0.02
2023	-0.31	-0.31	-0.33	-0.41
2062	-0.28	-0.30	-0.22	-0.38
2071	-0.16	-0.13	-0.28	-0.32
2087	-0.37	-0.33	-0.41	-0.59
2088	-0.53	-0.49	-0.48	-0.80
2089	-0.38	-0.46	-0.38	-0.60
2092	-0.24	-0.14	-0.22	-0.21
2104	-0.12	-0.16	-0.11	-0.13
2110	-0.26	-0.22	-0.11	-0.15
2129	-0.28	-0.31	-0.36	-0.25
2167	-0.17	-0.22	-0.22	-0.35
2202	-0.02	-0.84	-0.09	-0.02
2261	0.09	0.02	0.02	0.06
2266	-0.28	-0.22	-0.27	-0.30
Average	-0.20(0.03)	-0.25(0.03)	-0.23(0.03)	-0.27(0.03)

An important question relevant to this paper concerns how good are these different score functions at producing the right geometry. The orbital score function is considered important on the basis that PCIs are important and modelling them explicitly will aid in ligand binding. Table 2

and Table 3 suggest that it makes slight improvements in capturing the native pose on both *Talaris* and *Score12*, while maintaining discriminatory power. However, it is important to determine if this is due to improved capture of PCIs. It is also important to search for factors affecting the cases in which it performs poorly. In order to do this the native structures have been compared to the model with the best RMSD among the top 10 by score to see if the score function can recapitulate the PCIs between the ligand and the protein.

Figure 13: Ligand Binding Site of model 2110. A: the ligand site of the crystal structure, B: the ligand site of the orbital score function, C: the ligand site of *Score12*, D: the ligand site of *Talaris2013*, and E: the ligand site of *Talaris2014*. These are all model 2110.

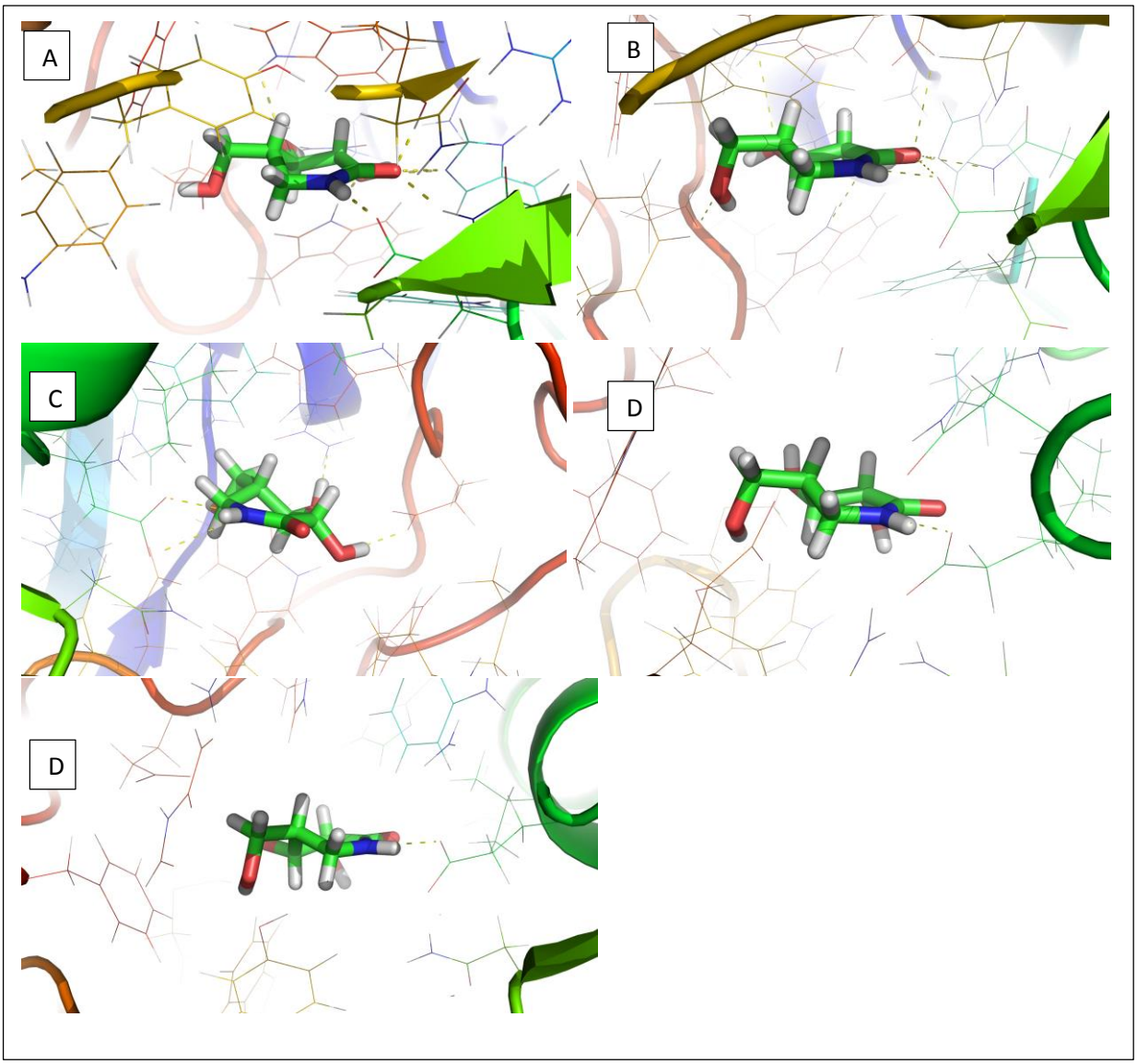


Table 4: Geometric properties of ligands in Figure 12.

	Dihedral	Distance	Angle 1	Angle 2
Crystal Structure (A)	143.58	2.07	141.27	148.72
Orbital (B)	127.73	1.68	163.36	149.54
Score12 (C)	56.6	4.4	100.76	105
Talaris2013(D)	-50.58	5.04	120.72	134.31
Talaris2014 (E)	116.3	4.01	73.43	68.73

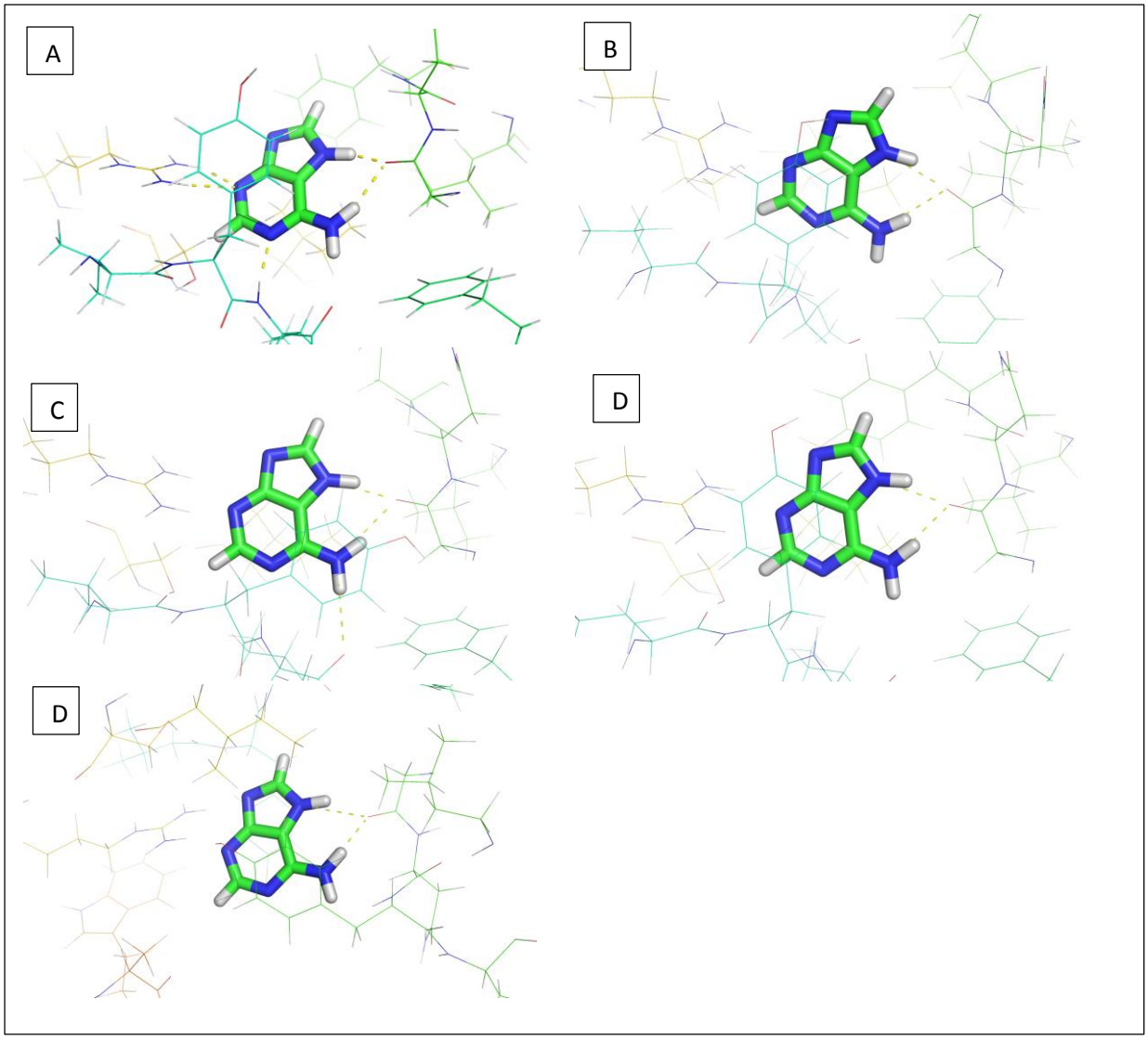
Table 4 and snapshots for model number 2110 in Figure 12 showcase the ligand for which the orbital score function performs better by a significant margin than the other three score functions. As can be seen from the ligand binding sites (Figure 12A), the orbital score function captures the ligand binding pocket correctly, maintaining the PCIs that are important for determining the orientation of the ligand in the binding pocket. These interactions are not captured by the other score functions, as noted especially by the distances of over 4 Å between the oxygen and the hydrogen that it is interacting with.

Among the models where all of the score functions perform well, the orbital score function continues to correctly capture the geometries of the PCIs. Model 2071 is shown Figure 13, specifically zoomed into an interacting hydrogen bond. The images show that each of the four score functions can properly capture the ligand in the ligand binding site. In both cases the orbital score function has dihedral angles that are somewhat poor. In the second case, the other score functions do a better job of finding the correct dihedral, with the exception of T*alaris2014*. Because the hydrogen-bonding statistics in *Talaris2013*, *score12* and *Talarais2014* are generated using the dihedral angle, and the orbital score function statistics are not, it is not surprising that they would perform slightly better than the orbital score function in this case. Angles one and two, however, are both used in the hydrogen bonding statistics that are used by the other three score functions. Despite the input angles being different in the orbital score function from the rest, the orbital score function performs very similarly to the others.

Table 5: Geometries of ligands in Figure 13.

	Dihedral	Distance	Angle 1	Angle 2
Crystal Structure	162.84	2.13	161.27	115.26
Orbital	149.53	1.98	165.83	110.3
Score12	168.07	2.1	165.17	112.32
Talaris2013	178.03	1.91	166.72	116.15
Talaris2014	149.78	1.97	149.01	129.2

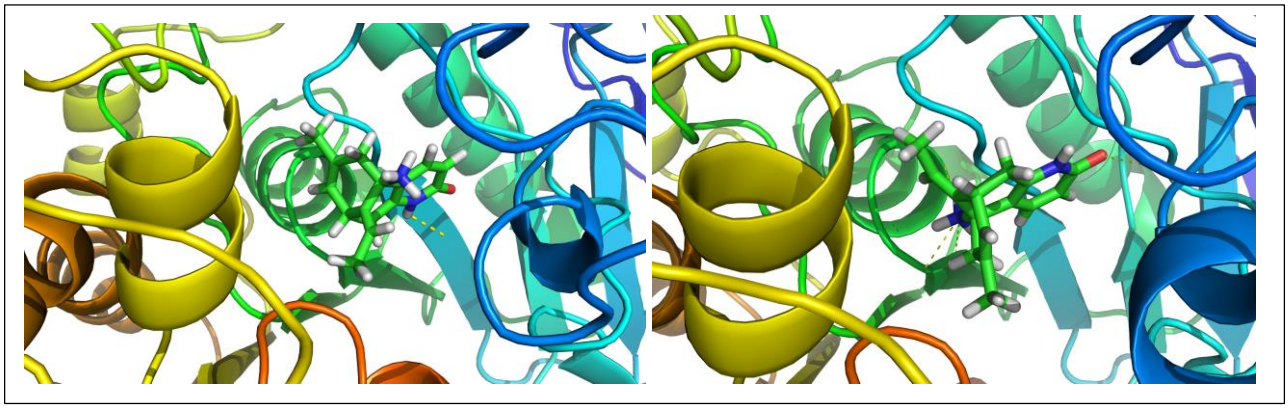
Figure 14: Ligand Binding site of Model 2071. A: the ligand site of the crystal structure, B: the ligand site of the orbital score function, C: the ligand site of *Score12*, D: the ligand site of *Talaris2013*, and E: the ligand site of *Talaris2014*. These are all model 2071.



Although the orbital score function performs well in many cases, we are going to consider two cases in which it did not. Models 1127 and 2087 are the cases in which the Orbital Score

Function performed the most poorly so they will be considered along with possible reasons for the poor performance.

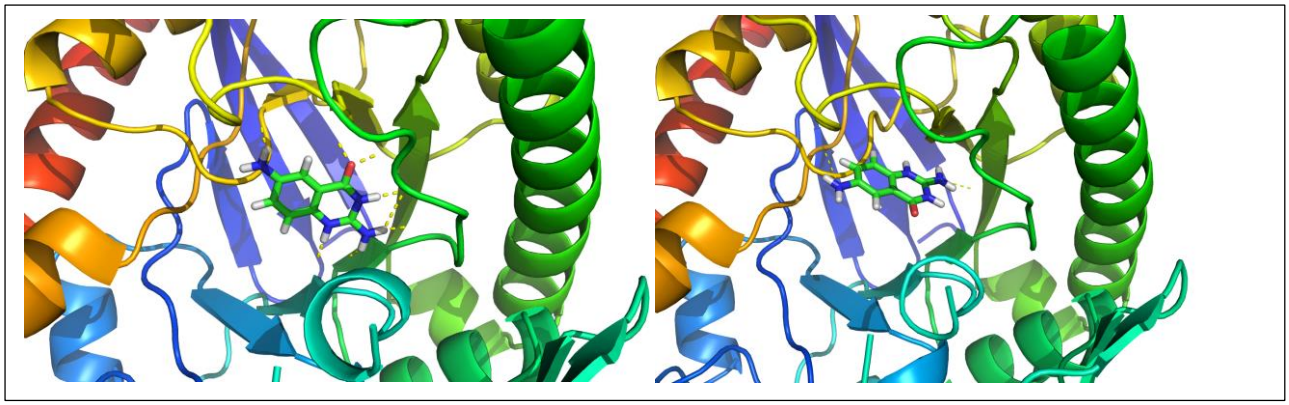
Figure 15: Model 1127 zoomed in on the ligand binding pocket. On the left is the crystal structure, the right is the result of the orbital score function.



The first model to be considered is model 1127. In this case the orbital score function has correctly identified the orientation and placement of the side chains around the binding pocket, but has incorrectly determined the orientation of the ligand. In this case the ligand is 180^{o} from what it is in the crystal structure. The reason that it has done this is in order to connect some PCIs from the amine group to side chains that are below. The ligand has very few opportunities for PCIs, as a large portion of the ligand contains sp3 hybridized carbons which do not contain orbitals that can undergo PCIs. The score function has identified possible interactions that do not exist in the crystal structure and given them good scores.

It is important to note that in this case the 180^{o} flip is very similar to the first, and it is possible that the ligand could fit in that orientation instead of the first inside the density map that is used to determine the structure. That is not how it was reported, but it is possible that that is the correct orientation. Also important to note is that there are models among the top 15 that have very good RMSD to native, suggesting that it does also consider the native structure to be favourable.

Figure 16: Model 2087 zoomed in on the ligand binding pocket. On the left is the crystal structure, the right is the result of the orbital score function.



The second model to consider is model 2087. In this case, many of the side chains are packed poorly and both the orientation and the placement of the ligand in the binding pocket are poor. Many of the PCIs that exist in the crystal structure still exist, but the side chains have moved in order to compensate for the movement of the ligand. While this might initially suggest either a score problem or potentially a sampling problem, the run in question did manage to produce sub angstrom models that maintain both the correct PCIs and the orientation and placement of the ligand in the binding pocket. This means that the correct conformation is being sampled, but the score that is used is scoring the non-native poses higher than the native ones.

Conclusions and Future Directions

A method to explicitly place orbitals and use them for scoring ligand docking has been created and implemented. QM knowledge has been exploited to intelligently place the orbitals on both the ligand and the protein. This method uses a combination of the hybridization state of the atom and the number of bonds to place both π orbitals and lone pairs, taking into account the environment to some extent. Each orbital consists of one point in space that is a distance away from the atom equal to the Van der Waals radius of the atom.

Statistics were generated for the types of interactions to develop a score term for Rosetta scoring function. This score term directly replaces the hydrogen bonding statistical potentials that

are used in the existing score functions. The other energy terms have been re-optimized for use with new score function AssignOrbitals.

This new score function based off of orbital placement directly captures PCIs instead of indirectly capturing hydrogen bonding interactions. Because PCIs are important in ligand docking it is expected that this score function will capture the pose of a ligand in the binding pocket better than the previous score functions. All three of the currently used score functions and the orbital score function were tested using a benchmark ligand docking set.

The orbital score function performed 10-15% better than the other score functions while maintaining or improving on the discrimination as determined by the funnel. This increase initially seems to be due to the high importance of PCIs in docking the ligand, and it recapitulates PCIs that are present between the ligand and the binding pocket. In the cases that it performs less well there are two general trends that are noticed. The first is that the protein has been folded properly and the ligand binding pocket is correct, but the orientation of the ligand has been shifted in order to produce PCIs that are not there in the crystal structure. In at least one case, it is even possible that the generated structure is correct, as it also would fit into the density map used to determine the crystal structure. The second trend that is seen is that the actual PCIs that exist are maintained intact, but the overall shape of the binding pocket is not maintained. In these cases the issue does not seem to be a sampling problem, but rather that the score function is correctly identifying the PCIs that are important but is unable to identify the proper orientation of the surrounding atoms correctly. It is possible that this is because some of the other terms are weighted poorly. In order to test this it would be important to re-optimize the other scoring terms that are used to see if it is possible to improve these cases while maintaining the other cases.

It is clear that there are still situations where improvements can be made, and one possible option would be to place multiple points for each orbital. This would create a larger set of statistics for each interaction, and would also potentially provide flexibility when choosing which point to interact with.

Another approach that is currently being used to determine orbitals is the QM-MM calculations on the whole protein-ligand complex. These calculations use Quantum Mechanics to determine the orbitals on a small section of the protein that is of interest, in this case the binding site, and Molecular mechanics on the rest of the system in order to take advantage of the information that is there. Because the greater portion of the protein is dealt with using molecular mechanics, the time constraint is not as big of an issue in these cases.

A trajectory can be computed for the binding event using QMMM calculations to examine the changes that the orbitals, on both the ligand and the protein in the binding pocket, undergo during binding event as the ligand settles into the site. This would also allow the use of other analysis tools, such as electron localization functions (ELFs)[30,31,32].

ELFs are used to analyse the results of QM calculation trajectories to determine the existence and location of interactions between ligand and the protein. They also determine electron density the location of interest. This type of analysis might lead to non-intuitive orbital placement as it gives locations where PCIs happen rather than areas of electron density.

Another way to improve on this method is to widen the benchmark set used to collect statics on PCIs. The benchmark set was chosen for the small size of the complexes, good resolution structures, and diversity among ligands. Because of this large protein families were not included, which is an area that Rosetta has sometimes struggled with. Examples include GPCR proteins, large antibodies, and protein-protein complexes that include a ligand. GPCRs especially are known to undergo large conformational changes between the bound and unbound state. In both cases, PCIs are often considered to be highly important in determining both the structure and the motion of the protein between both states. This score function seeks to take advantage of explicit orbitals and should perform best when applied to systems where structure is considered to be driven by the formation or breaking of PCIs. In order to test whether or not the current orbital implementation properly captures the existing PCIs, a benchmark run including this type of protein-ligand complex

would be important. However, limitation to setting up such a benchmark set is the lower resolution of such structures and the larger overall size of the complex.

Methods

The Quantum Mechanics Calculations were carried out using the Jaguar program of the Schrodinger Molecular Suite of Programs. Single Point energy calculations were carried out using Density Functional Theory (DFT)[33] with B3LYP functional[34] and LAVP2** basis set, which was chosen for its flexibility and accuracy. The other options considered for single point calculations include restricted SCF, medium grid density, atomic overlap initial guess and 50 maximum iterations for gradient minimization. Molecular orbitals upto HOMO-10 and LUMO+10 were calculated and exported to isomesh. PYMOL software (ref) was used for orbital visualization using isomesh levels +5 and -5.

Modelling of the electron orbitals was done as a single point at a distance equal to the Bohr radius, the most probable distance of an electron to the nucleus of an atom, with geometric parameters defined by atomic hybridization. Exceptions to this are the cases of ketenes and oxygens with a double bond to a sulphur or phosphorus. Atomic orbitals were defined as lone pair or bonding π orbitals. Three types of interactions were defined, with statistics for both the acceptor and donor ends of the interaction: bonding π - bonding π, bonding π - hydrogen, and lone pair - hydrogen.

The orbital energy function was derived based on geometrical parameters seen in protein crystal structures for partial covalent interactions. The geometric measurements used were a distance between the orbital and the hydrogen (δ_{HOrb}), an angle between the acceptor – orbital – hydrogen (AOH) ($\cos(\Psi)$), and an angle between the donor – hydrogen – orbital (DHO) ($\cos(\Theta)$). Inclusion of a direct measurement between the AHO and DHO angle results in a precise definition of the chemical interaction seen in PCIs and removes the need to indirectly calculate the relationship with torsion angles as seen with the hydrogen bond potential.[7]

For derivation of the knowledge based potential, the Rosetta Features Reporter[5] was used to obtain distances and angles representative of PCIs in the top8000 dataset (see supplemental for command lines). The inverse Boltzmann relation was used to convert the propensity of δ_{HOrb}, cos(Ψ) and δ_{HOrb}, cos(Θ) into an energy.

$$E(X) = -RT\ln(P_{observed}(X)/P_{background}(X))$$

where E(X) is the energy for X, the feature observed, R the gas constant, T the temperature and $P_{observed}(X)$ the probability of the feature observed and $P_{background}(X)$ is the probability of the given observation seen by chance. The total energy for a given PCI is determined by the combination of $E(PCI|\delta_{HOrb}, \cos(\Psi)) + E(PCI \mid \delta_{HOrb}, \cos(\Theta))$ where PCI is the partial covalent interaction being modeled and δ_{HOrb}, cos(Ψ) is the distance and acceptor – orbital – hydrogen (AOH) angle and δ_{HOrb}, cos(Θ)) is the distance and donor – hydrogen - orbital (DHO) angle.

PCI distributions were determined by the shortest distance (δ_{HOrb}) between two participating residues. Once the shortest distance was determined, the cosine of both Ψ and Θ were determined to account for bias in observing a given angle by chance. Two-dimensional histograms were created for both δ_{HOrb}, cos(Ψ) and δ_{HOrb}, cos(Θ) with bin fractions set to 0.1 Å for δ_{HOrb} and 0.1 for cos(Ψ) and cos(Θ)). The expected background probabilities for δ_{HOrb} were determined by dividing each bin fraction by the squared distance (r^2) for each observed bin fraction. Further, pseudo counts were added to each bin fraction to ensure that favorable states received a negative energy.

Although the shortest distance for PCIs was used to determine bin fractions, a bicubic interpolation of all distances for every PCI was used to determine the energy associated with a PCI between two residues. This has two direct effects, i) the energy function becomes a continuous function and ii) bicubic interpolation ensures that δ_{HOrb}, cos(Ψ) and δ_{HOrb}, cos(Θ) remain tightly coupled.

Calculation of free energy in Rosetta is done through a linear combination of weighted scoring terms. The base score function in Rosetta is composed of van der Waals interactions (*fa_atr*), an inter side chain (*fa_rep*) and intra side chain repulsive term (*fa_intra_rep*), an implicit solvation model (*fa_sol*), a hydrogen bond term for side chain – side chain (*hbond_sc*), backbone – backbone

(*hbond_sr_bb, hbond_lr_bb*) and backbone – side chain (*hbond_bb_sc*), a backbone – dependent rotamer probability (*fa_dun*), the probability of an amino acid given *phi* and *psi* angles (*p_aa_pp*), the probability of two polar residues being within a certain distance of each other (*fa_pair),* and reference energies to resemble the quantity of residues seen in any given protein (*ref*):

$$\Delta G = W_{atr}E_{atr} + W_{rep}E_{rep} + W_{intra\_rep}E_{intra\_rep} + W_{sol}E_{sol} + W_{hbond\_sc}E_{hbond\_sc} + W_{hbond\_sr\_bb}E_{hbond\_sr\_bb} + W_{hbond\_lr\_bb}E_{hbond\_lr\_bb} + W_{hbond\_bb\_sc}E_{hbond\_bb\_sc} + W_{dun}E_{dun} + W_{p\_aa\_pp}E_{p\_aa\_pp} + W_{pair}E_{pair} + W_{ref}E_{ref}$$

The relative weights for all scoring terms were parameterized on a high resolution structure dataset using a conjugate gradient method to maximize the probability of the native amino acid at each position in the protein. Addition and removal of scoring terms to the free energy calculation requires adjustment of the individual weights.

To account for all PCIs, the orbital score function was split into three distinct score terms for weight optimization 1) *orbitals_hpol* which contained T-stacked cation-π, salt bridges, and hydrogen bonds, 2) *orbitals_haro* which contained T-stacked and offset parallel π-π interactions, and 3) *orbitals_orbitals* which contained parallel π-π interactions and cation-π interactions. A feature of AssignOrbitals is the ability to implicitly capture interactions that are difficult to model; however, this feature can result in a bias for certain interactions. Within the Rosetta score function, the pair potential and the hydrogen bond potential both model separate interactions; however, the pair potential also captures hydrogen bonds, which results in double counting. In an attempt to remove double counting, both the sc_hbond, hbond_bb_sc, and fa_pair were removed. The new free energy calculation resulted as:

$$\Delta G = W_{atr}E_{atr} + W_{rep}E_{rep} + W_{intra\_rep}E_{intra\_rep} + W_{sol}E_{sol} + W_{orbitals\_hpol}E_{orbitals\_hpol} + W_{hbond\_sr\_bb}E_{hbond\_sr\_bb} + W_{hbond\_lr\_bb}E_{hbond\_lr\_bb} + W_{orbitals\_haro}E_{orbitals\_haro} + W_{dun}E_{dun} + W_{p\_aa\_pp}E_{p\_aa\_pp} + W_{orbitals\_orbitals}E_{orbitals\_orbitals} + W_{ref}E_{ref}$$

Because PCIs are dependent upon the solvated environment, the solvation potential needed to be adjusted. The particle swarm optimization algorithm, OptE, was used to optimize all orbital score terms, the solvation term, and the reference energies. Initial weights for the all optimized terms were then varied by 0.05 to 0.3 while the reference energies were allowed to be optimized by OptE. Each resulted weight set was tested against a barrage of tests.

Models were generated using RosttaScripts[35]. Each run was for 5000 models. Results are given for the funnels produced and for the best RMSD among the top ten models by score. The low resolution step was completed using translate, rotate, slide together.

Appendix

Assign Orbital Code

```
// -*- mode:c++;tab-width:2;indent-tabs-mode:t;show-trailing-whitespace:t;rm-trailing-spaces:t -*-
// vi: set ts=2 noet:
//
// (c) Copyright Rosetta Commons Member Institutions.
// (c) This file is part of the Rosetta software suite and is made available under license.
// (c) The Rosetta software is developed by the contributing members of the Rosetta Commons.
// (c) For more information, see http://www.rosettacommons.org. Questions about this can be
// (c) addressed to University of Washington UW TechTransfer, email: license@u.washington.edu.
#include <core/chemical/Atom.hh>
#include <core/chemical/ResidueType.hh>
#include <core/chemical/orbitals/AssignOrbitals.hh>
#include <map>
#include <core/chemical/ChemicalManager.hh>
#include <core/chemical/AtomTypeSet.hh>
#include <core/chemical/AtomType.hh>
#include <utility/vector1.hh>
#include <utility/string_util.hh>
#include <numeric/xyz.functions.hh>
#include <ObjexxFCL/string.functions.hh>
#include <numeric/conversions.hh>
#include <numeric/constants.hh>
#include <numeric/NumericTraits.hh>
#include <core/chemical/ResidueTypeSet.hh>
#include <core/chemical/gasteiger/GasteigerAtomTyper.hh>
#include <core/chemical/modifications/ValenceHandler.hh>
#include <core/chemical/gasteiger/GasteigerAtomTypeSet.hh>
#include <core/chemical/gasteiger/GasteigerAtomTypeData.hh>
#include <core/chemical/orbitals/OrbitalTypeSet.hh>
#include <core/chemical/orbitals/OrbitalType.hh>
#include <core/types.hh>
#include <core/chemical/ResidueType.hh>
#include <core/chemical/ResidueTypeSet.hh>
#include <numeric/xyzVector.hh>
#include <core/chemical/Elements.hh>
#include <ObjexxFCL/format.hh>
#include <basic/Tracer.hh>

namespace ObjexxFCL { namespace format { } } using namespace ObjexxFCL::format; // AUTO USING
NS

namespace core{
namespace chemical{
namespace orbitals{

basic::Tracer TR("core::chemical::orbitals::AssignOrbitals");
```

```
std::string AssignOrbitals::strip_whitespace( std::string const & name )
{
	std::string trimmed_name( name );
	ObjexxFCL::left_justify( trimmed_name ); ObjexxFCL::trim( trimmed_name ); // simpler way
to do this?
	return trimmed_name;
}

//function helps place double/triple bonds orbitals perpendicular to the plane.
utility::vector1< numeric::xyzVector< core::Real > > AssignOrbitals::perpendicular_orbitals_helper
(
		core::Size const & atm_index1,
		core::Size const & atm_index2,
		core::Size const & atm_index3
)
{


	//define two vectors, both pointing back to the central atom with atm_index2
	numeric::xyzVector< core::Real > vector_d( restype_->atom( atm_index1 ).ideal_xyz() -
restype_->atom( atm_index2 ).ideal_xyz() );
	numeric::xyzVector< core::Real > vector_f( restype_->atom( atm_index1 ).ideal_xyz() -
restype_->atom( atm_index3 ).ideal_xyz() );

	//Create an object of Class utility::vector1 to hold the xyz coordinates of orbitals(e.g., cross
products)
	//Get two cross products of the two vectors, one is above, the other is below the plane
defined by the two vectors
	utility::vector1< numeric::xyzVector<core::Real> > pi_orbital_xyz_vector;

	numeric::xyzVector< core::Real > xyz_right = cross_product( vector_d, vector_f );
	numeric::xyzVector< core::Real > xyz_left = cross_product( -vector_d, vector_f );

	//Normalize the two new vectors, xyz_right and xyz_left to get a unit vector.
	//pi_orbital_xyz_vector now stores the new xyz coordinates of the pi orbitals.
	pi_orbital_xyz_vector.push_back( ( xyz_right.normalized() * atom_orbital_distance_ ) +
restype_->atom( atm_index1 ).ideal_xyz() );
	pi_orbital_xyz_vector.push_back( ( xyz_left.normalized() *  atom_orbital_distance_ ) +
restype_->atom( atm_index1 ).ideal_xyz() );

	return pi_orbital_xyz_vector;
}

//function helps place double/triple bond orbitals parallel to the plane
utility::vector1< numeric::xyzVector < core::Real > > AssignOrbitals::parallel_orbitals_helper
(
		core::Size const & atm_index1,
		core::Size const & atm_index2,
```

```
		core::Size const & atm_index3
)
{
	//define two vectors, both pointing back to the central atom with atm_index2
	numeric::xyzVector< core::Real > vector_d( restype_->atom( atm_index1 ).ideal_xyz() -
restype_->atom( atm_index2 ).ideal_xyz() );
	numeric::xyzVector< core::Real > vector_f( restype_->atom( atm_index1 ).ideal_xyz() -
restype_->atom( atm_index3 ).ideal_xyz() );

	//Create an object of Class utility::vector1 to hold the xyz coordinates of orbitals(e.g., cross
products)
	//Get two cross products of the two vectors, one is above, the other is below the plane
defined by the two vectors
	utility::vector1< numeric::xyzVector<core::Real> > pi_orbital_xyz_vector;

	numeric::xyzVector< core::Real > xyz_right = cross_product( vector_d, vector_f );

	//Normalize the two new vectors, xyz_right and xyz_left to get a unit vector.
	//pi_orbital_xyz_vector now stores the new xyz coordinates of the pi orbitals.

	numeric::xyzVector< core::Real > xyz_up = cross_product( xyz_right,vector_d );
	numeric::xyzVector< core::Real > xyz_down = cross_product( -xyz_right,vector_d );

	pi_orbital_xyz_vector.push_back( ( xyz_up.normalized() * atom_orbital_distance_ ) +
restype_->atom( atm_index1 ).ideal_xyz() );
	pi_orbital_xyz_vector.push_back( ( xyz_down.normalized() *  atom_orbital_distance_ ) +
restype_->atom( atm_index1 ).ideal_xyz() );

	return pi_orbital_xyz_vector;
}

//function for handling ketenes, similar to perpendicular and parallel helper functions but changes
the order of operations
numeric::xyzVector< core::Real > AssignOrbitals::ketene_orbitals_helper
(
		core::Size const & atm_index1,
		core::Size const & atm_index2,
		core::Size const & atm_index3
)
{
	//define two vectors, both pointing back to the central atom with atm_index2
	numeric::xyzVector< core::Real > vector_d( restype_->atom( atm_index1 ).ideal_xyz() -
restype_->atom( atm_index2 ).ideal_xyz() );
	numeric::xyzVector< core::Real > vector_f( restype_->atom( atm_index1 ).ideal_xyz() -
restype_->atom( atm_index3 ).ideal_xyz() );

	//Create an object of Class utility::vector1 to hold the xyz coordinates of orbitals(e.g., cross
products)
```

```
	//Get two cross products of the two vectors, one is above, the other is below the plane
defined by the two vectors
	numeric::xyzVector< core::Real >  pi_orbital_xyz_vector;

	numeric::xyzVector< core::Real > xyz_right = cross_product( vector_d, vector_f );

	//Normalize the two new vectors, xyz_right and xyz_left to get a unit vector.
	//pi_orbital_xyz_vector now stores the new xyz coordinates of the pi orbitals.
	pi_orbital_xyz_vector = ( ( xyz_right.normalized() * atom_orbital_distance_ ) + restype_-
>atom( atm_index1 ).ideal_xyz() );

	return pi_orbital_xyz_vector;
}

//Calls perpendicular orbital placer to place Pz orbitals when then exist.
void AssignOrbitals::Pz_orbitals_placer
(
		core::Size const & atm_index1,
		core::Size const & atm_index2,
		core::Size const & atm_index3
)
{
	utility::vector1< numeric::xyzVector< core::Real > > pi_xyz_coords(
			perpendicular_orbitals_helper(
					atm_index1,
					atm_index2,
					atm_index3
			)

	);
	for ( core::Size orbitals=1; orbitals <=pi_xyz_coords.size(); ++orbitals ){
		calculate_orbital_icoor(
				pi_xyz_coords[orbitals],
				atm_index1,
				atm_index2,
				atm_index3,
				core::chemical::orbitals::bonding_pi,
				false
		);
	}
}

//Calls parallel orbital placer to place Px or Py orbitals when then exist.
void AssignOrbitals::Pxy_orbitals_placer
(
		core::Size const & atm_index1,
		core::Size const & atm_index2,
		core::Size const & atm_index3
```

```
)
{
	utility::vector1< numeric::xyzVector<core::Real> > pi_xyz_coords(
			parallel_orbitals_helper(
					atm_index1,
					atm_index2,
					atm_index3
			)
	);

	for (core::Size orbitals=1; orbitals <=pi_xyz_coords.size(); ++orbitals){
		calculate_orbital_icoor(
				pi_xyz_coords[orbitals],
				atm_index1,
				atm_index2,
				atm_index3,
				core::chemical::orbitals::bonding_pi,
				false
		);
	}
}

//function that places lone pairs on trigonal pyramidal atoms (eg: nitrogen with three bonds and one
lone pair)
void AssignOrbitals::trigonal_pyramidal_orbitals_placer
(
		core::Size const & atm_index1,
		core::Size const & atm_index2,
		core::Size const & atm_index3,
		core::Size const & atm_index4
)
{
	//This function is for placing the lone pair in a trigonal pyramidal geometry. Example: a
nitrogen with three single bonds.

	//define three vectors starting from the neighbors_ and pointing back to the central atom
	numeric::xyzVector < core::Real > vector_a = ( restype_->atom( atm_index1 ).ideal_xyz() -
restype_->atom( atm_index2 ).ideal_xyz() );
	numeric::xyzVector < core::Real > vector_b = ( restype_->atom( atm_index1 ).ideal_xyz() -
restype_->atom( atm_index3 ).ideal_xyz() );
	numeric::xyzVector < core::Real > vector_c = ( restype_->atom( atm_index1 ).ideal_xyz() -
restype_->atom( atm_index4 ).ideal_xyz() );

	//add the three vectors together making a vector pointing up from the nitrogen and then
normalize and multiply by distance.
	numeric::xyzVector < core::Real > vector_d = ( vector_a + vector_b + vector_c );
	utility::vector1 < numeric::xyzVector < core::Real > > lone_pair_coords;
```

```
	lone_pair_coords.push_back( vector_d.normalized() * atom_orbital_distance_ + restype_-
>atom( atm_index1 ).ideal_xyz() );
	for ( core::Size orbitals=1; orbitals <=lone_pair_coords.size(); ++orbitals ){
		calculate_orbital_icoor(
				lone_pair_coords[orbitals],
				atm_index1,
				atm_index2,
				atm_index3,
				core::chemical::orbitals::lone_pair,
				false
		);
	}
}

//Function that places orbitals for one bond sp instances.
void AssignOrbitals::nr_know_bonds_1_sp ( core::Size const current_atom, core::Size const
secondary_atom, core::Size const tertiary_atom )
{
	if ( non_bonding_lone_pair_orbitals_ == 0 ){ //Function that sets Pz, Py, Px normally.
		for(

	std::set<core::chemical::gasteiger::GasteigerAtomTypeData::AtomicOrbitalTypes>::const_ite
rator
				it =  pi_orbitals_.begin(); it != pi_orbitals_.end(); ++it
		){
			if( *it == core::chemical::gasteiger::GasteigerAtomTypeData::Pz ){
				Pz_orbitals_placer(
						current_atom,
						secondary_atom,
						tertiary_atom
				);
			}
			if( *it == core::chemical::gasteiger::GasteigerAtomTypeData::Py || *it ==
core::chemical::gasteiger::GasteigerAtomTypeData::Px ){
				Pxy_orbitals_placer(
						current_atom,
						secondary_atom,
						tertiary_atom
				);
			}
		}
	} else if ( non_bonding_lone_pair_orbitals_ == 1 ) {
		//Linear lone pair placer.
		calculate_orbital_icoor(
				core::chemical::modifications::linear_coordinates( restype_->atom(
current_atom ).ideal_xyz(), restype_->atom( secondary_atom ).ideal_xyz(), atom_orbital_distance_
),
				current_atom,
```

```
                secondary_atom,
                tertiary_atom,
                core::chemical::orbitals::bonding_pi,
                false
        );

        //Function that sets Pz, Py, Px normally.
        for(

    std::set<core::chemical::gasteiger::GasteigerAtomTypeData::AtomicOrbitalTypes>::const_ite
rator
                it =  pi_orbitals_.begin(); it != pi_orbitals_.end(); ++it
        ){
            if( *it == core::chemical::gasteiger::GasteigerAtomTypeData::Pz ){
                Pz_orbitals_placer(
                        current_atom,
                        secondary_atom,
                        tertiary_atom
                );
            }
            if( *it == core::chemical::gasteiger::GasteigerAtomTypeData::Py || *it ==
core::chemical::gasteiger::GasteigerAtomTypeData::Px ){
                Pxy_orbitals_placer(
                        current_atom,
                        secondary_atom,
                        tertiary_atom
                );
            }
        }
    } else if ( non_bonding_lone_pair_orbitals_ == 2 ) {
        //This actually doesn't work/make sense; exit.
        utility_exit_with_message("What? you have two lone_pair orbitals on sp
hybridization!");//sp should not have two pairs. Wrong arrangement
    }
}

void AssignOrbitals::nr_known_bonds_1_sp2_P_S_elements_case( core::Size const current_atom,
core::Size const secondary_atom, core::Size const tertiary_atom ){
    bool two_bonded_oxygens = false;
    for ( core::Size j = 1; j <= restype_->bonded_neighbor( secondary_atom ).size(); ++j )
    {
        if ( restype_->atom( restype_->bonded_neighbor( secondary_atom )[j]
).element_type()->element() == core::chemical::element::O && restype_->bonded_neighbor(
secondary_atom )[j] != current_atom )
        {
            Pz_orbitals_placer
            (
                    current_atom,
```

```
                                        secondary_atom,
                                        restype_->bonded_neighbor( secondary_atom )[j]
                                );
                                Pxy_orbitals_placer
                                (
                                        current_atom,
                                        secondary_atom,
                                        restype_->bonded_neighbor( secondary_atom )[j]
                                );
                                calculate_orbital_icoor
                                (
                                        core::chemical::modifications::linear_coordinates( restype_-
>atom( current_atom ).ideal_xyz(), restype_->atom( secondary_atom ).ideal_xyz(),
atom_orbital_distance_ ),
                                        current_atom,
                                        secondary_atom,
                                        tertiary_atom,
                                        core::chemical::orbitals::bonding_pi,
                                        true
                                );
                                two_bonded_oxygens = true;
                        }
                }
                if ( !two_bonded_oxygens )
                {
                        Pz_orbitals_placer
                        (
                                current_atom,
                                secondary_atom,
                                tertiary_atom
                        );
                        Pxy_orbitals_placer
                        (
                                current_atom,
                                secondary_atom,
                                tertiary_atom
                        );
                        calculate_orbital_icoor
                        (
                                core::chemical::modifications::linear_coordinates( restype_->atom(
current_atom ).ideal_xyz(), restype_->atom( secondary_atom ).ideal_xyz(), atom_orbital_distance_
),
                                current_atom,
                                secondary_atom,
                                tertiary_atom,
                                core::chemical::orbitals::bonding_pi,
                                true
                        );
```

```
	}
}

//function that works on ketenes when there is only one known bond but is sp2 hybridized
void AssignOrbitals::nr_known_bonds_1_sp2_ketene_case(core::Size const current_atom, core::Size
const secondary_atom, core::Size const tertiary_atom ){
	//This is the odd case of ketenes.
	//A ketene is a linear functional group that has a carbon double-bonded to a carbon that is
double bonded to an oxygen.
	//This linear set of double bonds works differently than normal ketones because the double
bonds sit in perpendicular planes.
	utility::vector1< core::Size > bonds_neighbor( restype_->bonded_neighbor( secondary_atom
) );
	utility::vector1< numeric::xyzVector< core::Real > > neighboring_positions;
	for( core::Size i=1; i <= bonds_neighbor.size(); ++i ){
		if( ( bonds_neighbor[i] != secondary_atom ) || ( bonds_neighbor[i] != current_atom
) ){
			neighboring_positions.push_back( restype_->atom(bonds_neighbor[i]
).ideal_xyz() );
			break;
		}
	}
	if( !neighboring_positions.empty() ){
		calculate_orbital_icoor
		(
			core::chemical::modifications::angle_coordinates(
				restype_->atom( current_atom ).ideal_xyz(),
				restype_->atom( secondary_atom ).ideal_xyz(),
				ketene_orbitals_helper(
					current_atom,
					secondary_atom,
					tertiary_atom
				),
				atom_orbital_distance_,
				120.0 / 180.0 * numeric::constants::d::pi,
				180.0 / 180.0 * numeric::constants::d::pi,
				false,
				numeric::xyzVector<core::Real>( 0.0)
			),
			current_atom,
			secondary_atom,
			tertiary_atom,
			core::chemical::orbitals::lone_pair,
			true
		);
	} else {
		calculate_orbital_icoor
		(
```

```
                        core::chemical::modifications::angle_coordinates(
                                restype_->atom( current_atom ).ideal_xyz(),
                                restype_->atom( secondary_atom ).ideal_xyz(),
                                ketene_orbitals_helper(
                                        current_atom,
                                        secondary_atom,
                                        tertiary_atom
                                ),
                                atom_orbital_distance_,
                                120.0 / 180.0 * numeric::constants::d::pi,
                                180.0 / 180.0 * numeric::constants::d::pi,
                                false,
                                numeric::xyzVector<core::Real>( 0.0)
                        ),
                        current_atom,
                        secondary_atom,
                        tertiary_atom,
                        core::chemical::orbitals::lone_pair,
                        true
                );
        }
        calculate_orbital_icoor
        (
                core::chemical::modifications::triganol_coordinates(
                        restype_->atom( current_atom ).ideal_xyz(),
                        restype_->atom( secondary_atom ).ideal_xyz(),
                        restype_->atom( tertiary_atom ).ideal_xyz(),
                        atom_orbital_distance_
                ),
                current_atom,
                secondary_atom,
                tertiary_atom,
                core::chemical::orbitals::lone_pair,
                true
        );
        for(

        std::set<core::chemical::gasteiger::GasteigerAtomTypeData::AtomicOrbitalTypes>::const_ite
rator
                it =  pi_orbitals_.begin(); it != pi_orbitals_.end(); ++it
        ){
                if(*it == core::chemical::gasteiger::GasteigerAtomTypeData::Pz){
                utility::vector1< numeric::xyzVector< core::Real > > pi_xyz_coords =
parallel_orbitals_helper
                        (
                                current_atom,
                                secondary_atom,
                                tertiary_atom
```

```
                                        );
                }
                if( *it == core::chemical::gasteiger::GasteigerAtomTypeData::Py || *it ==
core::chemical::gasteiger::GasteigerAtomTypeData::Px ){
                        utility::vector1< numeric::xyzVector< core::Real > > pi_xyz_coords =
perpendicular_orbitals_helper
                                        (
                                                current_atom,
                                                secondary_atom,
                                                tertiary_atom
                                        );
                }
        }

}

//Function that places orbitals for one bond sp2 instances. This function contains some of the more
complicated cases like ketenes and sulfur/phosphorus amines.
void AssignOrbitals::nr_know_bonds_1_sp2 ( core::Size const current_atom, core::Size const
secondary_atom, core::Size const tertiary_atom ){
        if (
                        restype_->atom( secondary_atom ).element_type()->element() ==
core::chemical::element::S ||
                        restype_->atom( secondary_atom ).element_type()->element() ==
core::chemical::element::P
        ) {
                nr_known_bonds_1_sp2_P_S_elements_case ( current_atom, secondary_atom,
tertiary_atom ); //solve for sulfur and phosphorus
                return;
        }

        if ( restype_->bonded_neighbor( secondary_atom ).size() == 2 ) {
                nr_known_bonds_1_sp2_ketene_case( current_atom, secondary_atom,
tertiary_atom ); //look at ketenes
                return;
        }

        if ( non_bonding_lone_pair_orbitals_ == 1 ) {
                //Function for linear lone pairs.
                calculate_orbital_icoor(
                                core::chemical::modifications::linear_coordinates( restype_->atom(
current_atom ).ideal_xyz(), restype_->atom( secondary_atom ).ideal_xyz(), atom_orbital_distance_
),
                                current_atom,
                                secondary_atom,
                                tertiary_atom,
                                core::chemical::orbitals::bonding_pi,
                                true
```

```
		);
		//Py, Pz, Px orbitals normally.
		for(
			std::set <
core::chemical::gasteiger::GasteigerAtomTypeData::AtomicOrbitalTypes >::const_iterator
			it =  pi_orbitals_.begin(); it != pi_orbitals_.end(); ++it
		){
			if(*it == core::chemical::gasteiger::GasteigerAtomTypeData::Pz) {
			Pz_orbitals_placer (
					current_atom,
					secondary_atom,
					tertiary_atom
			);
			}
			if( *it == core::chemical::gasteiger::GasteigerAtomTypeData::Py || *it ==
core::chemical::gasteiger::GasteigerAtomTypeData::Px ) {
			Pxy_orbitals_placer (
					current_atom,
					secondary_atom,
					tertiary_atom
			);
			}
		}
	} else if ( non_bonding_lone_pair_orbitals_ == 2 ) {

		utility::vector1< numeric::xyzVector<core::Real> > positions;

		positions.push_back(restype_->atom(current_atom).ideal_xyz() - (restype_-
>atom(tertiary_atom).ideal_xyz() - restype_->atom(secondary_atom).ideal_xyz() ).normalize() * 1 );
		positions.push_back( core::chemical::modifications::triganol_coordinates(
			restype_->atom(current_atom).ideal_xyz(),
			restype_->atom(secondary_atom).ideal_xyz(),
			positions[1],
			1.0
		)
		);

		for(core::Size ii=1; ii<= positions.size(); ++ii){
			calculate_orbital_icoor(
				positions[ii],
				current_atom,
				secondary_atom,
				tertiary_atom,
				core::chemical::orbitals::lone_pair,
				true
			);
		}
	} else if ( non_bonding_lone_pair_orbitals_ == 3 ) {
```

```
                //This doesn't really make sense. Shouldn't exist. Exit.
                utility_exit_with_message("What? you have three lone_pair orbitals on sp2
hybridization!");//this should never happen
        }
}

//Function that places orbitals for one bond sp3 instances
void AssignOrbitals::nr_know_bonds_1_sp3 ( core::Size const current_atom, core::Size const
secondary_atom, core::Size const tertiary_atom ) {
        //One bond sp3 only has to place the three sets of lone pair orbitals. Function here to do
that.

        numeric::xyzVector<core::Real > ketene_xyz = ketene_orbitals_helper(current_atom,
secondary_atom, tertiary_atom);
        numeric::xyzVector<core::Real > first_orbital_position =
core::chemical::modifications::angle_coordinates(
                        restype_->atom( current_atom ).ideal_xyz(),
                        restype_->atom( secondary_atom ).ideal_xyz(),
                        ketene_xyz,
                        atom_orbital_distance_,
                        120.0 / 180.0 * numeric::constants::d::pi,
                        180.0 / 180.0 * numeric::constants::d::pi,
                        false,
                        numeric::xyzVector<core::Real>( 0.0)
        );
        // coordinate 1
        calculate_orbital_icoor(
                        first_orbital_position,
                        current_atom,
                        secondary_atom,
                        tertiary_atom,
                        core::chemical::orbitals::lone_pair,
                        false
        );
        // helper coordinates
        numeric::xyzVector< core::Real > foot_point(
                        core::chemical::modifications::triganol_coordinates(
                                        restype_->atom( current_atom ).ideal_xyz(),
                                        restype_->atom( secondary_atom ).ideal_xyz(),
                                        first_orbital_position,
                                        atom_orbital_distance_ * std::cos( 54.75 / 180 *
numeric::constants::d::pi )
                        )
        );
        numeric::xyzVector< core::Real > offset(
                        atom_orbital_distance_ * std::sin( 54.75 / 180 * numeric::constants::d::pi ) *
                        cross_product(
```

```
                                        restype_->atom( secondary_atom ).ideal_xyz() - restype_->atom( current_atom ).ideal_xyz(),
                                        first_orbital_position - restype_->atom( current_atom ).ideal_xyz()
                        ).normalize()
        );
        // coordinate 2
        calculate_orbital_icoor(
                        foot_point + offset,
                        current_atom,
                        secondary_atom,
                        tertiary_atom,
                        core::chemical::orbitals::lone_pair,
                        false
        );
        // coordinate 3
        calculate_orbital_icoor(
                        foot_point - offset,
                        current_atom,
                        secondary_atom,
                        tertiary_atom,
                        core::chemical::orbitals::lone_pair,
                        false
        );
}

//Function that places orbitals for two bonds sp instances
void AssignOrbitals::nr_know_bonds_2_sp ( core::Size const current_atom, core::Size const secondary_atom )
{
        if ( non_bonding_lone_pair_orbitals_ == 0 ) {
                //Consider the case of =C= or -C#. We do not have orbitals to assign
                //but we do have p orbitals to assign.
                for(

        std::set<core::chemical::gasteiger::GasteigerAtomTypeData::AtomicOrbitalTypes>::const_iterator
                                        it =  pi_orbitals_.begin(); it != pi_orbitals_.end(); ++it
                ){
                        if( *it == core::chemical::gasteiger::GasteigerAtomTypeData::Pz ) {
                                Pz_orbitals_placer(
                                                current_atom,
                                                secondary_atom,
                                                neighbors_[2]
                                );
                        }
                        if( *it == core::chemical::gasteiger::GasteigerAtomTypeData::Py || *it == core::chemical::gasteiger::GasteigerAtomTypeData::Px ) {
```

```
                                Pxy_orbitals_placer(
                                        current_atom,
                                        secondary_atom,
                                        neighbors_[2]
                                );
                        }
                }
        }else if ( non_bonding_lone_pair_orbitals_ == 1 ) {
                //This really should be possible because it would require 10 electrons. Exit.
                utility_exit_with_message("What? You are trying to assign a lone pair to SP
hybridized atom with multiple bonds.");
        }
}
//Function that places orbitals for two bonds sp2
void AssignOrbitals::nr_know_bonds_2_sp2( core::Size const current_atom, core::Size const
secondary_atom, core::Size const tertiary_atom ) {
        if ( non_bonding_lone_pair_orbitals_ == 0 || non_bonding_lone_pair_orbitals_ == 1 ) {
                //Both of these cases add a set of bonding p orbitals. The second case also will add a
lone pair.
                for(

        std::set<core::chemical::gasteiger::GasteigerAtomTypeData::AtomicOrbitalTypes>::const_ite
rator
                                it =  pi_orbitals_.begin(); it != pi_orbitals_.end(); ++it
                ){
                        if(*it == core::chemical::gasteiger::GasteigerAtomTypeData::Pz) {
                                Pz_orbitals_placer(
                                        current_atom,
                                        secondary_atom,
                                        tertiary_atom
                                );
                        }
                        if( *it == core::chemical::gasteiger::GasteigerAtomTypeData::Py || *it ==
core::chemical::gasteiger::GasteigerAtomTypeData::Px ) {
                                Pxy_orbitals_placer(
                                        current_atom,
                                        secondary_atom,
                                        tertiary_atom
                                );
                        }
                }
                if ( non_bonding_lone_pair_orbitals_ == 1 ) {
                        calculate_orbital_icoor(
                                core::chemical::modifications::triganol_coordinates(
                                        restype_->atom( current_atom ).ideal_xyz(),
                                        restype_->atom( secondary_atom
).ideal_xyz(),
                                        restype_->atom( neighbors_[2] ).ideal_xyz(),
```

```
                                                atom_orbital_distance_
                                        ),
                                        current_atom,
                                        secondary_atom,
                                        tertiary_atom,
                                        core::chemical::orbitals::lone_pair,
                                        true
                                );
                        }
                } else if ( non_bonding_lone_pair_orbitals_ == 2 ) {
                        //This should never really happen or be possible. Exit.
                        utility_exit_with_message("What? You are trying to place two lone pairs and two
sets of bonds on an SP2 hybridized atom.");
                }
}

//Function that places orbitals for two bonds sp3
void AssignOrbitals::nr_know_bonds_2_sp3( core::Size const current_atom, core::Size const
secondary_atom )
{
        if ( non_bonding_lone_pair_orbitals_ == 1 ) {
                //Uncommon but possible in some cases. Place the one lone pair only in a trigonal
manner.
                calculate_orbital_icoor(
                                core::chemical::modifications::triganol_coordinates(
                                                restype_->atom( current_atom ).ideal_xyz(),
                                                restype_->atom( secondary_atom ).ideal_xyz(),
                                                restype_->atom( neighbors_[2] ).ideal_xyz(),
                                                atom_orbital_distance_
                                ),
                                current_atom,
                                secondary_atom,
                                neighbors_[2],
                                core::chemical::orbitals::lone_pair,
                                true

                );
        } else if ( non_bonding_lone_pair_orbitals_ == 2 ) {
                //This is the most normal case. Place the two lone pair in a tetrahedral manner.
                // helper coordinates
                numeric::xyzVector<core::Real> foot_point(
                                core::chemical::modifications::triganol_coordinates(
                                                restype_->atom( current_atom ).ideal_xyz(),
                                                restype_->atom( secondary_atom ).ideal_xyz(),
                                                restype_->atom( neighbors_[2] ).ideal_xyz(),
                                                atom_orbital_distance_ * std::cos( 54.75 / 180 *
numeric::constants::d::pi )
                                )
```

```
		);

		numeric::xyzVector < core::Real > offset(
			atom_orbital_distance_ * std::sin( 54.75 / 180 *
numeric::constants::d::pi ) * cross_product
			(
				restype_->atom( secondary_atom ).ideal_xyz() -
restype_->atom( current_atom ).ideal_xyz(),
				restype_->atom( neighbors_[2] ).ideal_xyz() -
restype_->atom( current_atom ).ideal_xyz()
			).normalize()
		);

		// coordinate 1
		calculate_orbital_icoor (
			foot_point + offset,
			current_atom,
			secondary_atom,
			neighbors_[2],
			core::chemical::orbitals::lone_pair,
			true
		);
		// coordinate 2
		calculate_orbital_icoor (
			foot_point - offset,
			current_atom,
			secondary_atom,
			neighbors_[2],
			core::chemical::orbitals::lone_pair,
			true
		);
	} else if ( non_bonding_lone_pair_orbitals_ == 3 ) {
		//Should not be possible or happen. Exit.
		utility_exit_with_message("What? You are trying to place three lone pairs on an
atom that already has at least 4 electrons");
	}
}

//Function that places orbitals for three bonds sp. This shouldn't be possible. Anything with this
many electrons should include d orbitals. Throws an error.
void AssignOrbitals::nr_know_bonds_3_sp( core::Size const current_atom ) {
	//This really should not be possible. Exit.
	utility_exit_with_message("What? You are trying to place three bonding sets on an SP
hybridized atom.");
}

//Function that places orbitals for three bonds sp2
```

```
void AssignOrbitals::nr_know_bonds_3_sp2( core::Size const current_atom, core::Size const
secondary_atom )
{
	if ( non_bonding_lone_pair_orbitals_ == 0 ) {

		//This is a case of an atom (carbon) with two single bonds and a double bond
		/*utility::vector1< numeric::xyzVector< core::Real > > pi_xyz_coords =
perpendicular_orbitals_helper (
					current_atom,
					secondary_atom,
					neighbors_[2]
		);
		for(core::Size ii=1; ii<= pi_xyz_coords.size(); ++ii){
			calculate_orbital_icoor(
						pi_xyz_coords[ii],
						current_atom,
						secondary_atom,
						neighbors_[2],
						core::chemical::orbitals::bonding_pi,
						false
			);
		}*/
		for(

	std::set<core::chemical::gasteiger::GasteigerAtomTypeData::AtomicOrbitalTypes>::const_ite
rator
					it =  pi_orbitals_.begin(); it != pi_orbitals_.end(); ++it
		){
			if( *it == core::chemical::gasteiger::GasteigerAtomTypeData::Pz ){
				Pz_orbitals_placer
				(
						current_atom,
						secondary_atom,
						neighbors_[2]
				);
			}
			if( *it == core::chemical::gasteiger::GasteigerAtomTypeData::Py || *it ==
core::chemical::gasteiger::GasteigerAtomTypeData::Px ){
				Pxy_orbitals_placer
				(
						current_atom,
						secondary_atom,
						neighbors_[2]
				);
			}
		}
	} else if ( non_bonding_lone_pair_orbitals_ == 1 ) {
		//This really should be possible. Exit.
```

```
                utility_exit_with_message("What? You are trying to put ten atoms around a
molecule of SP2 hybridization!");
        }
}

//Funcion that places orbtials for three bonds sp3
void AssignOrbitals::nr_know_bonds_3_sp3( core::Size const current_atom ) {
        if ( non_bonding_lone_pair_orbitals_ == 0 ) {
                //No orbitals to place or lone pairs. May not need this line of code at all.
        } else if ( non_bonding_lone_pair_orbitals_ == 1 ) {
                //There is only a lone pair to be placed here. Use a function to do that.
                trigonal_pyramidal_orbitals_placer(
                                current_atom,
                                neighbors_[1],
                                neighbors_[2],
                                neighbors_[3]
                );

        } else if ( non_bonding_lone_pair_orbitals_ == 2 ) {
                //This should not be possible. Exit.
                //utility_exit_with_message("What? You are trying make an SP3 hybridized atom
with two sets of lone pairs and three bonds!");
        }
}

//This function cycles through the atoms and places orbitals on them if they need to be placed.
//First it checks for the number of bonds the atom has. After that it checks hybridization on that
atom.
//Then it checks lone pairs. It places the lone pairs first and then hyrbidized orbitals based
//on normal rules for assigning chemical orbitals.

//Each time there is a utility exit with message it is because there are more than 8 electrons around
an atom center.
//While this can sometimes happen for ligands (mostly for sulphur and phosphorus) these cases
don't have have orbtials
//that need placed. These cases are being looked at through QM calculations to make sure that the
orbitals don't need placed.
void AssignOrbitals::assign_orbitals() {
        core::chemical::ChemicalManager* chemical_manager =
core::chemical::ChemicalManager::get_instance();
        core::chemical::AtomTypeSetCAP atom_type_set = chemical_manager-
>atom_type_set("fa_standard");

        //this code relies heavily on gasteiger atom types. Make sure to assign them before trying to
go through other functions
        core::chemical::gasteiger::assign_gasteiger_atom_types( *restype_,
core::chemical::ChemicalManager::get_instance()->gasteiger_atom_type_set(), /*keep_existing=*/
false );
```

```
	//somewhat weird to finalize the restype, but this is needed because we are going to check if the atom is a backbone atom. If it is
	//we dont want to add orbitals
	restype_->finalize();
	for( core::Size current_atom=1; current_atom <= restype_->natoms(); ++current_atom ) {
		//why do we have to check for the name of a dummy atom???? This is only for GB_AA_PLACEHOLDER. This atom should be typed as virtual!!!!!
		//for now, ignore backbone atoms...need to find a fix for this!
		if( restype_->is_virtual( current_atom ) || restype_->atom_name( current_atom ) == "DUMM" || restype_->atom_is_backbone(current_atom) ){

			continue;
		}

		core::chemical::AtomType const & atmtype( restype_->atom_type( current_atom ) );

		if( atmtype.atom_has_orbital() ) {
			//setup the distance for where the orbital will be placed. The distance is the covalent radius for what atom we are on
			for(core::Size ii= 1; ii <= restype_->bonded_neighbor_types( current_atom).size(); ++ii){
				if(restype_->bonded_neighbor_types( current_atom)[ii] == core::chemical::TripleBond){
					atom_orbital_distance_ = restype_->atom( current_atom).gasteiger_atom_type()->get_atom_type_property( gasteiger::GasteigerAtomTypeData::CovalentRadiusTripleBond);
					break;
				}
				else if(restype_->bonded_neighbor_types( current_atom)[ii] == core::chemical::DoubleBond){
					atom_orbital_distance_ = restype_->atom( current_atom).gasteiger_atom_type()->get_atom_type_property( gasteiger::GasteigerAtomTypeData::CovalentRadiusDoubleBond);
					break;
				}
				else if(restype_->bonded_neighbor_types( current_atom)[ii] == core::chemical::SingleBond){
					atom_orbital_distance_ = restype_->atom( current_atom).gasteiger_atom_type()->get_atom_type_property( gasteiger::GasteigerAtomTypeData::CovalentRadiusSingleBond);
					//dont break here, because we might have a double bond as well, and thats what the orbital distance will be!
				}
				//this is rather annoying. Aromatic bonds are actual bond order double or triple. For now, assume aromatic bond will
				//be bond order double and hope everything is all right
```

```
                                else if(restype_->bonded_neighbor_types( current_atom)[ii] ==
core::chemical::AromaticBond){
                                        std::cout << "found aromatic bond: " << restype_->name()
<< std::endl;

                                        atom_orbital_distance_ = restype_->atom(
current_atom).gasteiger_atom_type()->get_atom_type_property(
gasteiger::GasteigerAtomTypeData::CovalentRadiusDoubleBond);
                                        //I am not sure what an aromatic bond is...is it double?
                                } else {
                                        atom_orbital_distance_ = 1.0;
                                }
                        }\
                        neighbors_ = restype_->bonded_neighbor( current_atom );
                        core::Size secondary_atom = neighbors_[1];
                        core::Size tertiary_atom = 0;

                        for( core::Size i=1; i<= restype_->bonded_neighbor( secondary_atom ).size();
++i ) {
                                if( restype_->bonded_neighbor( secondary_atom )[i] !=
current_atom ){
                                        tertiary_atom = restype_->bonded_neighbor(
secondary_atom )[i];
                                        break;
                                }
                        }
                        if(tertiary_atom == 0) {
                                for( core::Size i=1; i<= restype_->bonded_neighbor( neighbors_[2]
).size(); ++i ){
                                        if( restype_->bonded_neighbor( neighbors_[2] )[i] !=
current_atom ){
                                                tertiary_atom = restype_->bonded_neighbor(
neighbors_[2] )[i];
                                                break;
                                        }
                                }
                        }

                        nr_known_bonds_ = neighbors_.size();

                        pi_orbitals_ = restype_->atom( current_atom ).gasteiger_atom_type()-
>get_binding_pi_orbitals();
                        non_bonding_lone_pair_orbitals_ = restype_->atom( current_atom
).gasteiger_atom_type()->get_number_hybrid_lone_pairs();
                        core::chemical::gasteiger::GasteigerAtomTypeData::HybridOrbitalType
orbital_type( restype_->atom( current_atom ).gasteiger_atom_type()->get_hybrid_orbital_type());
                        if ( nr_known_bonds_ == 1 )
                        {
```

```
                                if ( orbital_type == gasteiger::GasteigerAtomTypeData::SP)
                                {
                                        nr_know_bonds_1_sp( current_atom, secondary_atom,
tertiary_atom );
                                } else if ( orbital_type == gasteiger::GasteigerAtomTypeData::SP2)
                                {
                                        //test_sp2( current_atom, secondary_atom, tertiary_atom );
                                        nr_know_bonds_1_sp2( current_atom, secondary_atom,
tertiary_atom );
                                } else if ( orbital_type == gasteiger::GasteigerAtomTypeData::SP3)
                                {
                                        nr_know_bonds_1_sp3( current_atom, secondary_atom,
tertiary_atom );
                                }
                        } else if ( nr_known_bonds_ == 2 )
                        {
                                if (orbital_type == gasteiger::GasteigerAtomTypeData::SP)
                                {
                                        nr_know_bonds_2_sp( current_atom, secondary_atom );
                                } else if ( orbital_type == gasteiger::GasteigerAtomTypeData::SP2)
                                {
                                        nr_know_bonds_2_sp2( current_atom, secondary_atom,
tertiary_atom);
                                } else if ( orbital_type == gasteiger::GasteigerAtomTypeData::SP3)
                                {
                                        nr_know_bonds_2_sp3( current_atom, secondary_atom );
                                }
                        } else if ( nr_known_bonds_ == 3 )
                        {
                                if( orbital_type == gasteiger::GasteigerAtomTypeData::SP)
                                {
                                        nr_know_bonds_3_sp( current_atom );
                                } else if ( orbital_type == gasteiger::GasteigerAtomTypeData::SP2)
                                {
                                        nr_know_bonds_3_sp2( current_atom, secondary_atom );
                                } else if ( orbital_type == gasteiger::GasteigerAtomTypeData::SP3)
                                {
                                        nr_know_bonds_3_sp3( current_atom );
                                }
                        }
                }
        }

        restype_->finalize();
}

void AssignOrbitals::test_sp2(core::Size current_atom, core::Size  secondary_atom, core::Size
tertiary_atom) {
```

```
	utility::vector1< numeric::xyzVector<core::Real> > positions;

	positions.push_back(restype_->atom(current_atom).ideal_xyz() - (restype_-
>atom(tertiary_atom).ideal_xyz() - restype_->atom(secondary_atom).ideal_xyz() ).normalize() * 1 );
	positions.push_back( core::chemical::modifications::triganol_coordinates(
			restype_->atom(current_atom).ideal_xyz(),
			restype_->atom(secondary_atom).ideal_xyz(),
			positions[1],
			1.0
	)
	);

	for(core::Size ii=1; ii<= positions.size(); ++ii){
		calculate_orbital_icoor(
				positions[ii],
				current_atom,
				secondary_atom,
				tertiary_atom,
				core::chemical::orbitals::lone_pair,
				true
		);
	}
	utility::vector1< numeric::xyzVector< core::Real > > pi_xyz_coords =
perpendicular_orbitals_helper
			(
					current_atom,
					secondary_atom,
					tertiary_atom
			);

	for( core::Size ii=1; ii<= pi_xyz_coords.size(); ++ii ){
		calculate_orbital_icoor(
				pi_xyz_coords[ii],
				current_atom,
				secondary_atom,
				tertiary_atom,
				core::chemical::orbitals::bonding_pi,
				false
		);
	}

}

std::string AssignOrbitals::make_orbital_element_name()
{
	++n_orbitals_;
	std::string orbital_name("X");
```

```
	std::string orb_index_string = utility::to_string < core::Size > ( n_orbitals_) ;
	std::string orbital_element_name(orbital_name+orb_index_string);
	return orbital_element_name;
}

void AssignOrbitals::set_orbital_type_and_bond(
		core::Size atom_index,
		std::string orbital_element_name,
		orbitals::OrbitalTypeEnum orbital_enum

){
	// Orbital names are given by concatenate two strings:'LP" and the indices of the orbitals on
the residue(restype_);
	std::string atm_name( strip_whitespace( restype_->atom_name(atom_index) ) );

	restype_->add_orbital( orbital_element_name, orbital_enum );
	restype_->add_orbital_bond( atm_name, orbital_element_name );

}

//This function calculates the internal coordinates of the orbitals that have been placed and adds
them to the residue type for further use.
void AssignOrbitals::calculate_orbital_icoor(
		numeric::xyzVector < core::Real > const orbital_xyz,
		core::Size const atm_index1,
		core::Size const atm_index2,
		core::Size const atm_index3,
		core::chemical::orbitals::OrbitalTypeEnum orbital_type,
		//bool here because the angle needs to be divided by two for one bond sp2 and not
in other cases
		bool theta_over_2
)
{
	core::chemical::AtomType const & atmtype( restype_->atom_type( atm_index1 ) );

	std::string orbital_element_name ( make_orbital_element_name() );
	set_orbital_type_and_bond( atm_index1, orbital_element_name, orbital_type );

	Vector const stub1_xyz = restype_->atom(atm_index1).ideal_xyz();
	Vector const stub2_xyz = restype_->atom(atm_index2).ideal_xyz();
	Vector const stub3_xyz = restype_->atom(atm_index3).ideal_xyz();

	core::Real theta(0.0);
	core::Real phi(0.0);

	if(atom_orbital_distance_ <1e-2)
	{
```

```
                TR << "WARNING: extremely small distance=" << atom_orbital_distance_ << " for "
<<
                                orbital_element_name << " ,using 0.0 for theta and phi."<<
                                " If you were not expecting this warning, something is very wrong"
<<std::endl;
        }else
        {
                theta = numeric::angle_radians<core::Real>(orbital_xyz,stub1_xyz,stub2_xyz);
                if( (theta < 1e-2) || (theta > numeric::NumericTraits<Real>::pi()-1e-2) )
                {
                        phi = 0.0;
                }else
                {
                        phi =
numeric::dihedral_radians<core::Real>(orbital_xyz,stub1_xyz,stub2_xyz,stub3_xyz);
                }
        }
        std::string const stub1(strip_whitespace(restype_->atom_name(atm_index1)));
        std::string const stub2(strip_whitespace(restype_->atom_name(atm_index2)));
        std::string const stub3(strip_whitespace(restype_->atom_name(atm_index3)));

        //tr << orbital << " " << stub_atom1 << " "<< stub_atom2 << " " <<stub_atom3 << " "
<<distance << " " << phi << " " << theta <<std::endl;
        if ( !theta_over_2 )
        {
                restype_->set_orbital_icoor_id( orbital_element_name, phi, theta,
atom_orbital_distance_, stub1, stub2, stub3);
        } else
        {
                restype_->set_orbital_icoor_id( orbital_element_name, phi, (theta/2),
atom_orbital_distance_, stub1, stub2, stub3);
        }
}
}//namespace
}//namespace
}//namespace
```

Sample XML

```
<ROSETTASCRIPTS>
        <SCOREFXNS>
                <ligand_soft_rep weights="%%soft_rep%%"/>
                <hard_rep weights="%%hard_rep%%"/>
        </SCOREFXNS>
        <TASKOPERATIONS>
                <ExtraRotamersGeneric name=extra_chi ex1=1 ex2=1 extrachi_cutoff=0/>
        </TASKOPERATIONS>
        <LIGAND_AREAS>
```

```
		<docking_sidechain chain=X cutoff=6.0 add_nbr_radius=true all_atom_mode=true
minimize_ligand=10/>
		<final_sidechain chain=X cutoff=6.0 add_nbr_radius=true all_atom_mode=true/>
		<final_backbone chain=X cutoff=7.0 add_nbr_radius=false all_atom_mode=true
Calpha_restraints=0.3/>
	</LIGAND_AREAS>
	<INTERFACE_BUILDERS>
		<side_chain_for_docking ligand_areas=docking_sidechain/>
		<side_chain_for_final ligand_areas=final_sidechain/>
		<backbone ligand_areas=final_backbone extension_window=3/>
	</INTERFACE_BUILDERS>
	<MOVEMAP_BUILDERS>
		<docking sc_interface=side_chain_for_docking minimize_water=true/>
		<final sc_interface=side_chain_for_final bb_interface=backbone
minimize_water=true/>
	</MOVEMAP_BUILDERS>
	<MOVERS>
		<ddG name=calculateDDG jump=1 per_residue_ddg=1 repack=0
scorefxn=hard_rep/>
		<Translate name=translate chain=X distribution=uniform angstroms=2.5 cycles=50/>
		<Rotate name=rotate chain=X distribution=uniform degrees=360 cycles=1000/>
		<SlideTogether name=slide_together chains=X/>
		<HighResDocker name=high_res_docker cycles=6 repack_every_Nth=3
scorefxn=ligand_soft_rep movemap_builder=docking/>
		<FinalMinimizer name=final scorefxn=hard_rep movemap_builder=final/>
		<InterfaceScoreCalculator name=add_scores chains=X scorefxn=hard_rep/>
		<ParsedProtocol name=low_res_dock>
			<Add mover_name=translate/>
			<Add mover_name=rotate/>
			<Add mover_name=slide_together/>
		</ParsedProtocol>
		<ParsedProtocol name=high_res_dock>
			<Add mover_name=high_res_docker/>
			<Add mover_name=final/>
		</ParsedProtocol>
	</MOVERS>
	<PROTOCOLS>
		<Add mover_name=low_res_dock/>
		<Add mover_name=high_res_dock/>
		Add mover_name=calculateDDG/>
		<Add mover_name=add_scores/>
	</PROTOCOLS>
</ROSETTASCRIPTS>
```

Sample PBS script

```
#!/bin/sh
#PBS -l nodes=1:ppn=1
```

```
#PBS -l pmem=5500mb
#PBS -l mem=5500mb
#PBS -l walltime=24:00:00
#PBS -o relax.log
#PBS -j oe

cd $DIR
/dors/meilerlab/home/willcotc/my_rosetta_stuff/ligand_docking/ligand/pbs/rosetta_scripts.static.linuxgccrelease -database /dors/meilerlab/home/willcotc/rosetta_database/ @/dors/meilerlab/home/willcotc/my_rosetta_stuff/ligand_docking/ligand/options/full_sample.txt -s "'$STRUCTURE $LIGAND'" -ex1 -ex2 -ex1aro -ex2aro -linmem_ig 10 -extra_res_fa $PARAMS -parser:script_vars hard_rep=$HARD soft_rep=$SOFT -add_orbitals -out:pdb_gz -nstruct 500 -in:file:native $NATIVE
```

Sample Submit Script

```
#! /bin/csh
#
# Sends respective protein/ligand pairs to ACCRE
#
foreach file (/dors/meilerlab/home/willcotc/my_rosetta_stuff/ligand_docking/ligand/input/brittany/pdb/*.pdb)
        set direct = `echo $file |  awk '{print "/dors/meilerlab/home/willcotc/my_rosetta_stuff/ligand_docking/ligand/output/ligand_full_sample/" substr($1,89, length($1)-92 ) }'`
        set name = `echo $file | awk '{print "fs_" substr($1,89)}'`
        set ligand = `echo $file | awk '{print "/dors/meilerlab/home/willcotc/my_rosetta_stuff/ligand_docking/ligand/input/brittany/ligand/" substr($1,89)}'`
        set params = `echo $file | awk '{print "/dors/meilerlab/home/willcotc/my_rosetta_stuff/ligand_docking/ligand/input/brittany/params/no_conformers/" substr($1,89 , length($1)-92) ".params"}'`
        set hard_rep = `echo ligand`
        set soft_rep = `echo ligand_soft_rep`
        set native = `echo $file |  awk '{print "/dors/meilerlab/home/willcotc/my_rosetta_stuff/ligand_docking/ligand/input/brittany/pdb/hetatm/" substr($1,89 , length($1)-92) ".pdb"}'`
#       echo $ligand
#       echo $name
        qsub -N ${name} -v DIR=${direct},STRUCTURE=${file},LIGAND=${ligand},PARAMS=${params},SOFT=${soft_rep},HARD=${hard_rep},NATIVE=${native} ../pbs/full_sample.pbs

end
```

Sample options input

```
#
-options
 -user
```

```
#
#
#-ignore_unrecognized_res
-add_orbitals
#
#
-parser
 -protocol
/dors/meilerlab/home/willcotc/my_rosetta_stuff/ligand_docking/ligand/options/full_sample.xml
#
#
-mute protocols.jd2
-mute core.io.pdb.file_data
-mute core.scoring.etable
-mute core.io.database
-mute core.scoring.ScoreFunctionFactory
-mute core.pack.task
-mute protocols.ProteinInterfaceDesign.DockDesign
```

References

1. Hajduk, P., Greer, J., A decade of fragment-based drug design: strategic advances and lessons learned. *Nature Reviews*. **6**:211-219 (2007).

2: Druker, B., ST1571(Gleevec) as a paradigm for cancer therapy. *Trends in Mol Med*. Vol 8 (**4**): S14-S18 (2002).

3. Dudek, A., Adroz, T., Galvez, J. Computational methods in developing quantitative structure-activity relationships (QSAR): A review. *Comb Chem High Throughput Screen.* Vol 9(**3**):213-228 (2006).

4. Adcock, S., McCammon, J., Molecular Dynamics: A survey of methods for simulating the activity of proteins. *Chem Rev*., **106**:1589-1615 (2006).

5: Ewing, T.J., Makino, S., Skillman, A.G., Kuntz, I.D., DOCK 4.0: search strategies for automated molecular docking of flexible molecule databases. *J Comp Aided Mol Des* 2001;15:411-428.

6: Osterberg, F., Morris, G.M., Sanner, M.F., Olson, A.J., Goodsell, D.S., Automated docking to multiple target structures: incorporation of protein mobility and structural water heterogeneity in Auto-Dock. *Proteins*. 2002; 46;34-40.

7: Rarey, M., Kramer, B., Lenguaer, T., Klebe, G., A fast flexible docking method using an incremental construction algorithm. *J Mol Biol* 1996;261: 470-489.

8: Willet, P., Glen, R.C., Leach, A.R., Taylor, R., Jones, G., Development and validation of a genetic algorithm for flexible docking. *J Mol Biol*. 1997;267:727-748.

9: Meiler, J., Baker, D., ROSETTALIGAND: Protien-Small molecule docking with full side-chain flexibility. *Proteins*. 2006;65:538-548.

10: Rohl, C.A, Strauss, C.E.M., Misura, K.M.S & Baker, K. Protein structure prediction using Rosetta. *Methods Enzymol*. **383**, 66-93 (2004).

11: T. Olsson, S. Bowden. An overview of protein-ligand docking using GOLD. EMBO workshop 2014.

12: AutoDock wiki. Autodock.com.

13: The FlexX Method. www.uku.fi/~poso/index_files/OOHJE.pdf.

14. Williams, D., Stephens, E., O'Brien, D., Zhou, M. Understanding noncovalent interactions: Ligand Binding Energy and Catalytic Efficiency from Ligand-Induced Reductions in Motion within Receptors and Enzymes. *Angew Chem Int Ed*., **43**: 6596-6616 (2004).

15. Lindhorst, T., Artificial Multivalent Sugar Ligands to Understand and Manipulate Carbohydrate-Protein Interactions. *Topics in Current Chemistry*. **218**:201-235 (2001).

16. Simons, K., Bonneau, R., Ruczinski, I., Baker, D. Ab initio structure prediction of CASP III targets using ROSETTA. *Proteins*. **37**: 171-176 (1999).

17: Leaver-Fay, A., O'Meara, M., et al. Scientific Benchmarks for Guiding Macromolecular Energy Function Improvement. *Methods Enxymol.* **523** 109-143 (2013).

18: Shapovalov, M.V., Dunbrack, R.L Jr., A smoothed backbone-dependent rotamer library for proteins derived from adaptive kernel density estimates and regressions. *Structure*. 2011; 19(6):844-858.

19: Song, Y., Tyka, M., Leaver-Fay, A., Thompson, J., Baker, D., Structure-guided forcefield optimization. *Proteins* 2011; 79:1898-1909.

20. Hunter, C. The Role of Aromatic Interactings in Molecular Recognition. Medola Lecture. 1994

21. Pollino, J., Weck, M. Non-covalent side-chain polymers: design principles, functionalization strategies, and perspectives. *Chem Soc Rev*. **34**: 193-207 (2005).

22: Kortemme, T., A.V. Morozov and D. Baker. An orientation-dependent hydrogen bonding potential improves prediction of specificity and structure for proteins and protein-protein complexes. *J Mol Biol* 326(4): 1239-59 (2003).

23: Keedy, D.A., Arendall III, W.B., Chen, V.B., Williams, C.J., Headd, J.J., Echols, N., et al. 8000 filtered structures, 2012 http://kinemage.biochem.duke.edu/databases/top8000.php.

24: Allison, B., Combs, S., DeLuca, S., Lemmon, G., Mizoue, L., Mieler, J., Computational Design of protein-small molecule interfaces. *Jou Struc Biol*. 2014; 193-202.

25: Gasteiger, J., Marsili, M., Iterative partial equalization of orbital electronegativity – a rapid access to atomic charges. *Tetrahedron* 1980; 3219-3288(36).

26: Griffiths, D., *Introduction to Quantum Mechanics*, Prentice-Hall 1995, p. 137. ISBN 0-13-124405-1.

27: Singh, S., Thornton, J., *Pi-Pi* interactions: the Geometry and Energetics of Phenylalanine-Phenylalanine Interactions in Proteins. *J. Mol. Biol.* 1991:837-846(218).

28. Combs, S., DeLuca, S., DeLuca, S., Lemmon, G., Nannemann, D., Nguyen, E., Willis, J., Sheehan, J., Meiler, J. Small-Molecule Ligand Docking into comparative models with Rosetta. *Nature Protocols*. **8**:1277-1298 (2013).

29: Conway, P., Tyka, M., DiMaio, F., Konerding, D., Baker, D., Relaxation of backbone bond geometry improves protein energy landscape modelling. *Protein Sci.* Vol 23:47-55 (2014).

30: Silvi, B., Savin, A., Classification of Chemical Bonds Based on Topological Analysis of Electron Localization Functions. *Nature.* Vol 371:683-686 (1994).

31: Becke, A., Edgecombe, K., A Simple Measure of Electron Localization in Atomic and Molecular Systems. *J Chem Phys*. **92**, 5397 (1990).

32. Savin, A., Nesper, R., Wengert, S., Fassler, T., ELF: The Electron Localization Function. *Angewantdte Chemie.* **36**:1808-1832 (1997).

33. Pierre, H., Kohn, W. Inhomogeneous electron gas. *Physical Review* **136**:867-871 (1964).

34: Kim, K., Jordan, K. Comparison of Density Functional and MP2 Calculations of the Water Monomer and Dimer. *J Phys Chem*. **98**:10089-10094 (1994).

35. Fleishman, S., Leaver-Fay, A., Corn, J., Strauch, E., Khare, S., Koga, N., Ashworth, J., Murphy, P., Richter, F., Lemmon, G., Meiler, J., Baker, D., RosettaScripts: A Scripting Language Interface to the Rosetta Macromolecular Modelling Suite. *Plos One*. **6**:1-10 (2011).

36. Huang, H., Lee, K., Yu, H., Chen, C., Hsu, C., Chen, H., Tsai, F., Chen, C. Structure-based and Ligand-based drug design for HER 2 Receptor. *Jou Bio Structure and Dynamics*. Vol 28(**1**) (2010).