Control of aggregated bacterial communities through engineered surface displayed proteins.

Thesis by
Christopher Johnstone

In Partial Fulfillment of the Requirements for the
Degree of
Bachelor of Science in Chemical Engineering

CALIFORNIA INSTITUTE OF TECHNOLOGY
Pasadena, California

2019

Christopher Johnstone
ORCID: 0000-0002-7255-0218

ACKNOWLEDGEMENTS

I would like to thank my graduate student mentor Mark Kozlowski, for teaching and guiding me throughout my time as an undergraduate researcher. Mark has been a wonderful role model and general source of advice, as well as freely giving encouragement and suggestions when I needed them.

I would like to thank Dave Tirrell for allowing me to join the lab that has been my research home and for giving me invaluable advice and suggestions on both research and graduate schools.

For all of my research time at Caltech, I'd like to thank all of the Tirrell lab members for their support and advice throughout.

For specific help on this thesis, I would like to thank Maiko Obana and Samuel Ho for helping me plan and execute the required organic syntheses. I also thank the Beckman Imaging Facility and the Beckman Institute Laser Resource Center for instrument support for the confocal microscopy images and circular dichroism data.

ABSTRACT

Bacterial aggregation through surface display of cross-associating proteins has previously been demonstrated, but the formation of these aggregates is only controllable and reversible through the addition of chemical inducers or soluble proteins. Here, we present a design for a photoswitchable surface-display system that causes bacterial aggregation. This system should reversibly disaggregate under exposure to blue light. We created our mutant by modifying Photoactive Yellow Protein (PYP), a fluorescent protein that undergoes a large reversible conformational change when exposed to blue light. We computationally designed this mutant to drive photoswitchable sequestration of a cap domain that is designed to selectively aggregate with SynZip18. Characterization of this designed protein's photoactivity was inconclusive due to its limited solubility though the synthesized chormophore, a p-coumaric acid derivative, was capable of reconstituting native photoactive PYP. While we did not show that the designed mutant could cause disaggregation under exposure to blue light, it was capable of selectively aggregating with surface displayed SynZip18 as desired.

TABLE OF CONTENTS

LIST OF ILLUSTRATIONS

LIST OF TABLES

Number *Page*

Chapter 1

INTRODUCTION

Natural bacterial communities, often seen in the form of biofilms, perform a diverse set of functions. As seen in Figure 1.1, these communities are versatile; by combining different strains and species of bacteria, favorable local environmental conditions can be created by bacteria interacting mutualistically. One species may protect another species from otherwise toxic molecules, or two species might exchange feed substrates. Bacterial communities also allow for specialization, where cells differentiate and perform different tasks for the community as a whole [1].

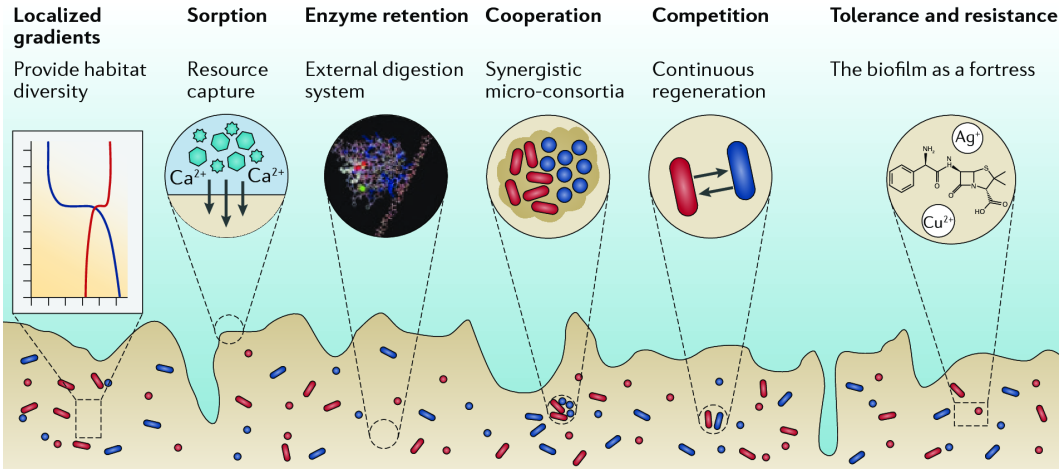


Figure 1.1: Emergent properties of biofilms. In nature, bacteria can form mutually beneficial communities where bacteria can take on specialized functions that aid in the growth and survival of the community. Figure adapted from Figure 1 of Flemming *et al.* [2].

Human-designed bacterial aggregates can recreate the mutual and complex interactions seen in natural bacterial communities. Creating such a system could allow novel engineering solutions. For example, the Ismagilov group used microfluidic extrusion to create an artificial community between two different bacteria—one species, *Sphingobium chlorophenolicum*, could degrade pentachlorophenol and another species, *Ralstonia metallidurans*, could reduce mercury [3]. These two pollutants often co-occur, but the mercury is toxic to the *S. chlorophenolicum*. By creating a shell of *R. metallidurans* around the *S.*

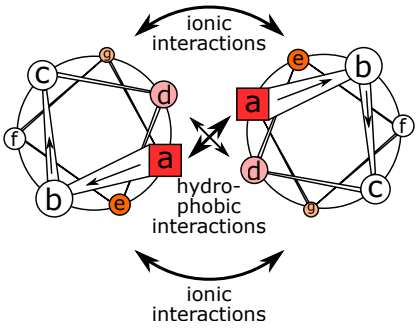
chlorophenolicum, the inner bacteria could be protected from the presence of mercury. Other artificial bacterial communities could be formed in order to simplify biosynthetic processes. Work in the Turner lab has shown that biocatalysis via a multi-enzyme cascade can create useful molecules that would otherwise be difficult or costly to synthesize; one such enzyme cascade was found to perform amine alkylation via in situ aldehyde formation, a reaction scheme that is difficult to perform synthetically [4]. However, one of the intermediate steps in the cascade can result in side products if performed under aerobic conditions. If this biosynthetic process was scaled up, the process would likely be designed with two separate bioreactors, one operating under aerobic conditions and one operating under anaerobic conditions. In order to maintain these conditions, surrounding separator and regulation units (for example, to control the percentage of dissolved oxygen), could also be required. If we could precisely control bacterial aggregation, a core-shell bacterial community could be formed in a single bioreactor, with the core bacteria experiencing a locally anaerobic environment, potentially greatly reducing the cost and complexity of the system.

Previous work in the Tirrell lab has demonstrated the ability to form bacterial aggregates via surface display of cross-associating proteins [5]. We accomplish surface display of target aggregation proteins by translationally fusing proteins of interest to a secretion tag and the autotransporter (AT) system. Autotransporter inserts into the cellular membrane and is normally responsible for secreting proteins into the extracellular environment. However, because our aggregation protein is translationally fused to autotransporter, it remains tethered to the surface of the bacteria [6]. Some aggregation proteins of interest include the SynZip proteins, leucine zippers that show cross- or self-association. These are interesting targets for surface display because their interactions are non-covalent and thus reversible [7, 8]. SynZip17 and SynZip18 were previously chosen as aggregation targets because they show strong cross-association but minimal self-association; this system was validated to cause aggregation in *Escherichia coli* through the autotransporter system. Furthermore, the reversibility of SynZip-based aggregation was demonstrated through the addition of soluble SynZip17; by disrupting the coiled-coil interactions, the bacterial aggregates drastically reduced in size.

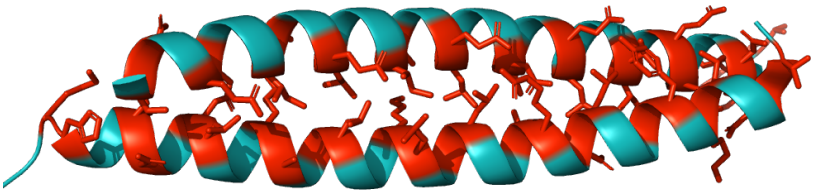
However, previous methods of controlling aggregate size are generally not time-resolved or easily repeatable. Addition of soluble protein to disrupt aggregation would make it difficult to re-aggregate the system without diluting out the added protein; use of varying amounts of chemical inducer or genetic control of expression levels is similarly inflexible and could be costly when scaled up, especially if large quantities of inducer would need to be added. Using a photoswitchable system, where light could be used to directly affect protein association, would be valuable and would allow precise temporal control of aggregation and disaggregation.

Several photoswitchable protein systems have been previously demonstrated. One such well-characterized system is the LOV2 system, which was derived from a light-oxygen-voltage sensitive protein sensor [9]. This system operates by sequestering an alpha helix in the dark state. In the light state, the bound helix is released and is able to interact with other proteins without steric hindrance. While this system would be useful for triggering transient aggregation, our work here seeks to design a system with transient disaggregation. Furthermore, the LOV2 alpha helix does not show significant sequence similarity to the SynZip system, meaning that the helix-core interface would have to be heavily redesigned. These considerations led us to reject this system from consideration. One similar photoswitch is based on Photoactive Yellow Protein (PYP). This protein normally undergoes a conformational change when exposed to blue light, where the beta strands around the photocenter unravel, causing a secondary conformational change in the N-terminal cap region, the region of the protein synthesized first by a ribosome. When the light source is removed, the protein switches back to its original conformation within seconds. This system can also be engineered to create dynamic protein behavior in response to light. In two key studies, DNA-binding domains replaced the N-terminus cap, resulting in a system that could selectively bind DNA unless exposed to blue light [10, 11].

Especially inspired by the work by Ali and coworkers [11], we were interested in adapting the PYP system in order to create a photoswitchable SynZip aggregation system. Based on the structural similarity between the SynZip proteins and CREB, we chose to engineer a mutant of PYP that replaced the N-terminus cap with a modified version of SynZip17 (SZ17).



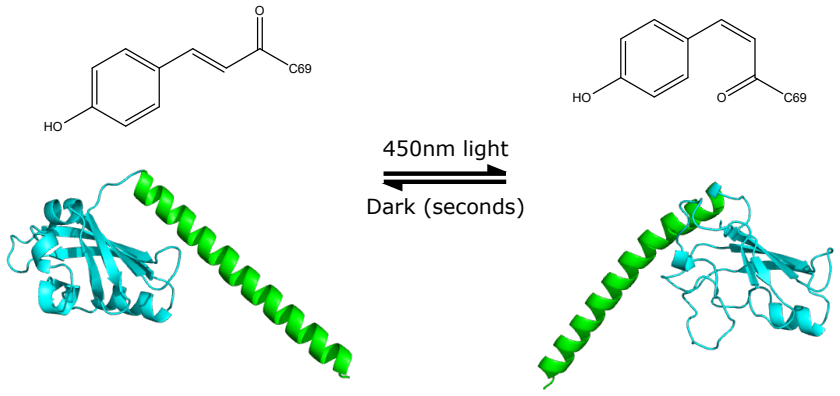
(a) Heptad repeat structure of coiled coils. Leucine zipper coiled coils follow a repeating seven-residue structure. The interacting hydrophobic and ionic residues can induce cross-association. Figure modified from Figure 1 of Mason & Arndt [12].



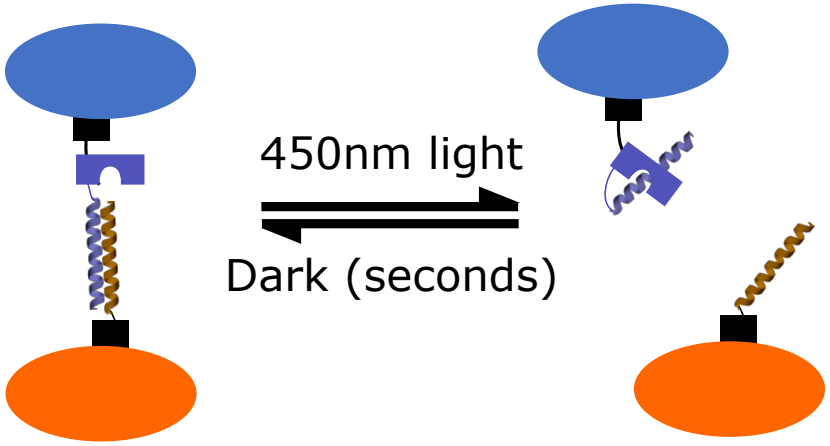
(b) Structure of two associated SynZip leucine zippers. Key interacting residues (the a/d/e/g heptad locations) are shown in red, whereas residues not involved in this interaction (the b/c/f heptad locations) are shown in teal.

Figure 1.2: Interaction of leucine zippers. Leucine zippers, including the SynZip family of proteins, are coiled coil domains that contain leucine at the d heptad location. In our computational methods, we designed for selective SynZip interactions by maintaining the sidechain identity of the key a/d/e/g residues, while allowing redesign of the b/c/f residues.

Using computational modeling techniques, we predicted the sequence of a SZ17-PYP mutant that should selectively sequester SZ17 upon irradiation by blue light, while allowing this SZ17 to interact with its complement, SZ18, in the dark state. We then created several genetic constructs in order to test the aggregation behavior of these proteins both in purified form and when expressed as part of an aggregation system in cells. The proposed system is shown in schematic form in Figure 1.3.



(a) Desired photoactive PYP mutant. Upon exposure to blue light, the p-coumaric thioester attached to a key cysteine residue undergoes a trans-cis isomerization. This causes a change in the protein secondary structure, exposing residues that favorably interact with the designed cap region.



(b) Illustration of the surface-displayed aggregation system.

Figure 1.3: Illustration of the desired final aggregation system. The autotransporter protein, shown in the illustration with black, remains tethered to the cell membrane and stays attached to the target protein of interest. The SZ-PYP fusion, normally exposes the SZ17-homologous cap region such that it can aggregate with cells surface-displaying SZ18. Upon irradiation with blue light, the PYP part of the protein sequesters the cap region, reversibly disrupting aggregation.

Chapter 2

COMPUTATIONAL DESIGN AND ANALYSIS OF A PHOTOSWITCHABLE AGGREGATION PROTEIN

2.1 Computational design

Computational design of the photoswitchable aggregation protein was performed with Rosetta, open-source software initially written by the Baker lab. Rosetta uses a combination of coarse centroid-based modeling of protein residues and detailed full-atom simulations to predict protein folding energies based on sequence and structure [13]. Rosetta's design tools have been extended to perform extensive backbone and sidechain remodeling [14], which we use here to optimize the light-state behavior of the mutant.

We split design of the photoactive mutant into two main stages: a N-terminal cap design phase where we redesigned the native PYP cap region to have SynZip17-like properties, followed by a light-state optimization phase where we enhanced stability and docking of the cap region to the core protein in the light state. Because our mutant replaces the cap region of a mutant PYP protein with a cap region with SynZip properties, our designed mutant is a combination of a SynZip-like cap (SZ) and a mutant version of PYP without the normal cap region (Δc-mPYP). In short, we call this mutant SZ-Δc-mPYP.

The second design phase, which encourages stable docking of the cap region to the core of the protein in the transient light state, was performed to inhibit aggregation in the light state. We posited that the introduction of strong interdomain binding between the cap and core in the light state would cause additional steric hindrance to the interprotein interactions necessary for aggregation.

SynZip cap redesign

For the first design phase, we followed a similar initial engineering strategy as used by Ali and coworkers [11]. Ideally, redesign of the cap could be performed over the entire statistical structural ensemble, where local optima could be discovered even among a large variation in both sidechain identity, sidechain orientation, and backbone orientation. However, current computa-

tional tools are not powerful enough to simultaneously consider the entire energy landscape. Thus, even though we expect the key SynZip17-like cap region to take on a structure between that of the native PYP N-terminal cap and that of the unencumbered leucine zipper, the initial design phase was done relative to the native dark-state PYP crystal structure (PDB 1NWZ). As shown in Figure 2.1, three types of residues were identified in the native PYP structure and optimized independently in this design phase.

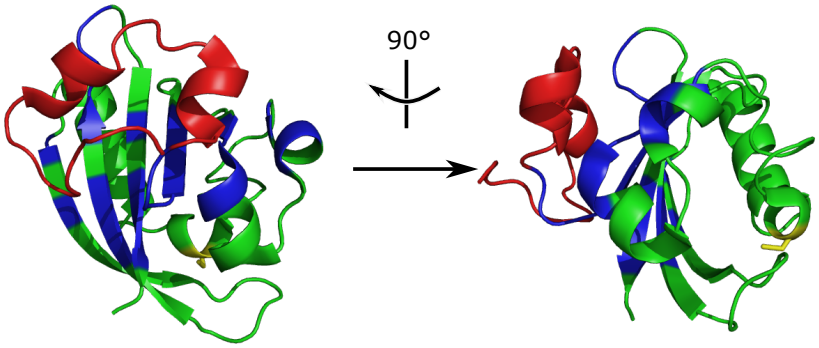


Figure 2.1: Computational mutagenesis strategy. The native PYP structure is shown, color-coded with the N-terminal cap redesign strategy. The red residues are the native PYP N-terminal cap that will be completely replaced with the SZ17-derived domain. The blue domains are all of the residues within 5 angstroms of the cap; these form the "binding pocket" that is redesigned to account for any unfavorable interactions, such as exposure of hydrophobic residues to solvent, created by modifications to the cap region. Green residues were not redesigned in the first N-terminal cap design phase. The chromophore is covalently attached to the residue shown in yellow, residue C69.

First, all of the residues of the cap region, shown in red in Figure 2.1, were fully redesigned (e.g. residue identity, sidechain orientation, and backbone position were allowed to change), using SynZip17's sequence to generate constraints on this region. Because of the heptad repeat structure seen in Figure 1.2, we enforced the identity of the SynZip17 residues at the a/d/e/g heptad locations in order to maintain favorable hydrophobic interactions between our SZ-Δc-mPYP protein and native SynZip18. The b/c/f residues were allowed to take on any identity (see constraint file and design script in Appendix B.1 and B.2). Explicit constraints against helix-breaking residues were not added at this stage, though the Rosetta redesign algorithm naturally selects against mutants that incur large energy costs.

Secondly, the amino acids within five angstroms of any of the cap residues, which we call the “SynZip binding pocket”, were redesigned while keeping their backbone position fixed. This redesign step accounts for any unfavorable interactions created by modification of the cap structure. For example, redesigning the cap into the SZ17-like domain can expose previously-buried hydrophobic residues on the beta sheet that the cap normally sits on. By redesigning the binding pocket region, we accounted for the destabilizing modifications made to the cap region. The other residues (green and yellow) were not redesigned. Finally, putative mutant proteins were minimized and "relaxed" (a method in Rosetta that performs full-atom refinement by searching local conformation space for the most energetically favorable structure) and scored. The mutant with the highest predicted stability was selected for further computational modeling.

Light-state optimization

We then selectively increased the stability and binding of the SZ-like cap in the light state. Starting with the same light-state and dark-state NMR-ensemble structures (PDB 1XFQ and 1XFN) for a capless version of native PYP (Δc-nPYP), we designed favorable binding between the mutant cap region and the rest of the protein through a docking-like design method. First, the mutant selected in the cap redesign step was remodeled (e.g. had the mutant sequence threaded onto a new initial backbone structure) using both the crystal structure of SynZip5, another leucine zipper around the same length as SynZip17, and the light-state Δc-nPYP structure.

After this remodeling step, we examined small angular perturbations between the N-terminal cap and the rest of the mutant protein, effectively sampling a space of possible “docked” configurations where the cap region tightly interacts with the rest of the protein. After minimization and relaxation steps performed to locate local minima in the energy landscape, the SynZip binding pocket was given a second opportunity to be redesigned. Through this process, additional stabilizing mutations in the binding site region were identified. After a final relaxation step, the most stable mutant was selected as SZ-Δc-mPYP (protein sequence available in Appendix A.1).

Decoy clustering and computational verification

In order to computationally verify that our designed mutant has the correct behavior, Rosetta was used to generate nearly four thousand structures ("decoys") of SZ-Δc-mPYP under different small angular perturbations between the SZ cap and the rest of the protein for both the light and dark states. While Rosetta naturally explores much of the local structural sample space while running design procedures, we found it helpful to fix the final mutant design and perform this type of decoy modeling as a check that the structural design space was sufficiently explored. In order to tractably examine the generated decoys, we implemented a modified version of Lloyd's algorithm (using average atomic root mean square distance as the metric; Python code available in Appendix C) and used it to cluster these decoys into 100 mean structures for both the light and dark states. The predicted Rosetta energy of each of these decoys was then averaged to return a cluster average folding energy. As shown in Figure 2.2, the most energetically favorable structures in the dark state show relatively small perturbations to the SZ17 leucine zippers, implying that aggregation could be possible. On the other hand, the most energetically favorable structures in the light state show extensive disruption and bending of the alpha-helix leucine zippers and favorable interactions with the core of the protein, likely indicating that aggregation would be disrupted by this enhanced light-state stability.

In order to predict if the designed mutant could interact favorably and selectively with SynZip18 for aggregation purposes, the bZIP prediction system created by the Singh lab was used to score the interactions between the designed SynZip cap region and SZ18. This prediction system used a large database of experimental data to train a Support Vector Machine model for the selectivity of arbitrary coiled coil interactions [15, 16].

As seen in Table 2.1, restricting the a/d/e/g heptad residues to the native SynZip17 residues is predicted to have maintained the selectivity of the desired SynZip interactions.

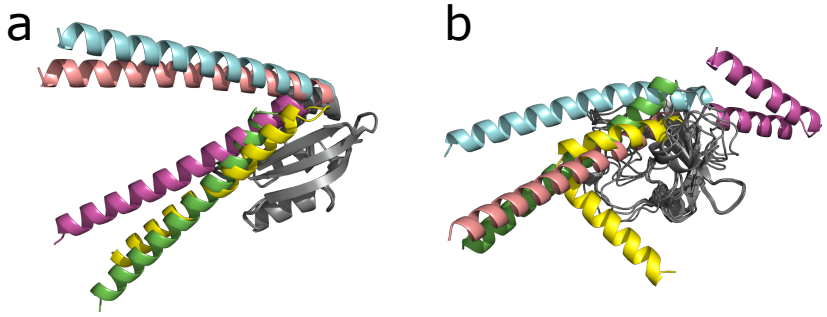


Figure 2.2: Predicted mutant structures in both the light and dark state. **a:** The five most energetically favorable clustered structures are shown for SZ-PYP in the dark state, with the core PYP region shown in gray. In the dark state, the SynZip cap region shows small amounts of perturbation, with some states barely interacting with the PYP core region. **b:** The five most energetically favorable clustered structures are shown for SZ17-PYP in the light state. These structures show fairly large disruptions in the normal leucine zipper structure, possibly indicating that this designed mutant will tightly sequester the SZ17 domain upon exposure to blue light.

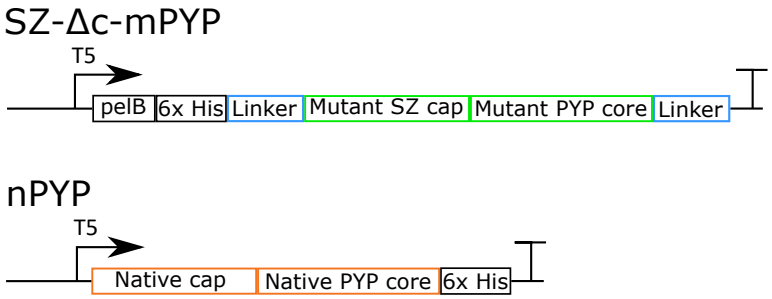
Coiled coil pair	SVM model score	Interaction percentile score
SynZip17 + SynZip17	-1.007	50.61
Designed cap + Designed cap	-14.2	7.32
Designed cap + SynZip18	26.564	96.65
SynZip17 + SynZip18	33.208	100.00

Table 2.1: Predicted selectivity of coiled-coil interactions. The interaction percentile score gives an estimation of the selectivity of the relevant interaction. A high percentile score means that this paired interaction is stronger than the interaction between all literature coiled coils in the database and the second pair member. For example, the designed cap + SynZip18 interaction is more favorable than 96.65% of the interactions between database coiled coils and SynZip18. While the designed SynZip cap is not as selective towards SynZip18 as SynZip17 is, the predicted selectivity between our designed cap region and SynZip18 is still relatively high, indicating that this computationally designed mutant has the possibility of enabling selective aggregation. Furthermore, our designed cap does not show strong self-interactions, indicating a predicted lack of self-aggregation.

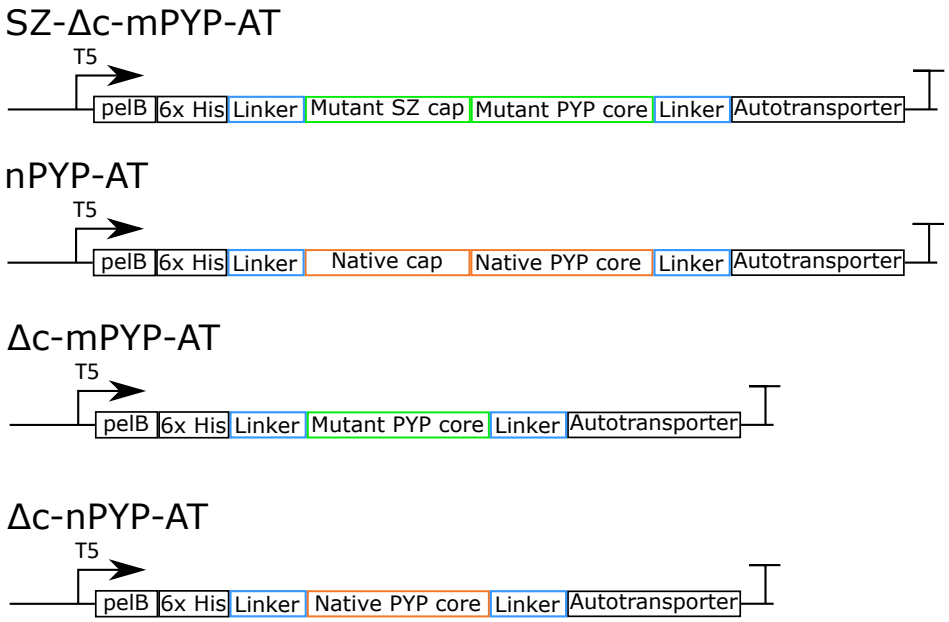
When comparing the predicted mutant to both SynZip17 and the non-cap region of native PYP, our computational model predicts 23 mutations in the non-cap region relative to the native PYP structure, in addition to 11 predicted mutations at the b/c/f heptad positions relative to the native SynZip17 structure. This large number of mutations could be detrimental to the photoactivity of the mutant, so we examined the seven residues that either form hydrogen bonds with or contribute to hydrophobic packing of the chromophore [17]. The predicted mutant maintains all of these residue identities except for the replacement of tyrosine 54 with tryptophan.

2.2 Genetic constructs

Six constructs were cloned in order to elucidate if the designed mutant has the desired aggregation and photoactivity properties; all six constructs also had a 6xHis tag added. Two constructs—the designed mutant, SZ-Δc-mPYP,(along with surrounding linker regions) and the native PYP protein, nPYP—were designed to be directly expressed, as seen in Figure 2.3a. Four constructs—the designed mutant and the native PYP protein in both capless and normal forms—were designed for surface display by combining these with the relevant autotransporter and secretion tag domains, as seen in Figure 2.3b.



(a) Constructs for expression. Constructs for both native PYP, modified with a His tag, and for the Rosetta-designed mutant plus linker regions, a His tag, and the secretion tag were created.



(b) Constructs for surface display. Four constructs were created to test surface display. The designed mutant and native PYP sequences, in both normal and capless forms, are translationally fused with the Autotransporter domain, a His tag, and a secretion tag.

Figure 2.3: Genetic constructs used for expression and aggregation experiments. Six constructs were all His-tagged and placed under the control of an IPTG-inducible promoter, T5. A series of six glycine/serine repeats was used as a flexible linker region. These constructs were placed on pQE80, a ColE1 origin plasmid with relatively high copy number. See Appendix A for full sequences.

2.3 Chromophore synthesis

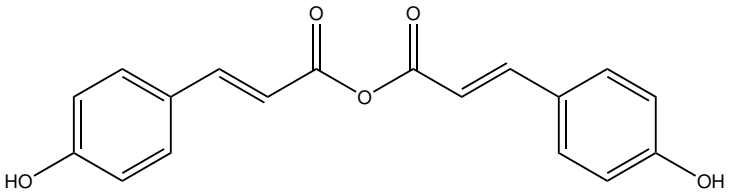
Native PYP derives its photoactive behavior from a chromophore, p-coumaric acid, that is covalently attached through a thioester linkage to a key cysteine residue (C69). Because *E. coli* do not have an optimized pathway to create this intermediate, PYP-based proteins synthesized under strong promoters in *E. coli* need to be supplemented with this chromophore. Previous work has shown that an activated version of p-coumaric acid (either a thioester or an anhydride) can either be added into the expression media or added directly to purified protein; either method results in photoactive PYP [18–20]. When anhydride is added to a protein solution, reconstitution occurs quickly, but the anhydride also quickly hydrolyzes in protic solvents like water, making it infeasible for use in bacterial culture. In contrast, the thioester hydrolyzes at a much slower rate, at the cost of an increased time until full reconstitution (sometimes up to hours).

In order to investigate the behavior of our mutant in comparison to the native PYP protein, we synthesized both p-coumaric anhydride and thiophenyl coumarate. For purified protein experiments, the anhydride form was used to quickly reconstitute photoactive PYP derivatives. For aggregation experiments, the thioester was used for stability reasons; as the cells grew in thioester-supplemented media, surface displayed protein could reconstitute as it was transported to the surface of the cells due to the slow hydrolysis rate.

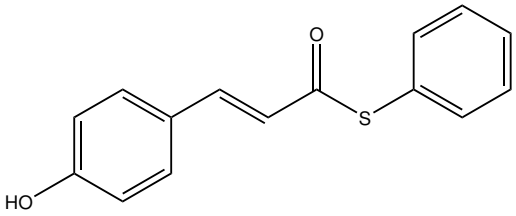
The first compound, p-coumaric anhydride, was synthesized through a dehydration reaction of p-coumaric acid; see section 4.4 for reaction scheme details. The thioester was synthesized in two different ways; one method used the synthesized anhydride and underwent a direct esterification reaction, whereas the other proceeded from the p-coumaric acid starting material; see Section 4.5 and 4.6 for reaction scheme details.

Purification of the resulting compounds was challenging. Likely due to the fast hydrolysis reaction of the anhydride, the anhydride was only isolated in a mixture with p-coumaric acid. The thiophenyl coumarate was similarly isolated with p-coumaric acid impurities under both reaction schemes. See sections 4.4, 4.5, and 4.6 for details.

However, because the presence of some p-coumaric acid would not necessarily interfere with the formation of the key thioester linkage, these synthesized



(a) Structure of p-coumaric anhydride.



(b) Structure of thiophenyl coumarate.

Figure 2.4: Activated chromophore structures. Either the anhydride or thioester derivative of p-coumaric acid can react with residue C69 in PYP to form photoactive PYP mutants.

products were used for further modification and aggregation experiments.

2.4 Protein modification

Expression and purification

The two constructs for protein expression as seen in Figure 2.3a were expressed at the 2L scale and purified via Ni-NTA column. Both proteins expressed at around 5mg protein per liter of culture, though both native PYP and our designed SZ-Δc-mPYP showed solubility problems when placed in native buffer solutions. During dialysis, protein visibly precipitated out of solution.

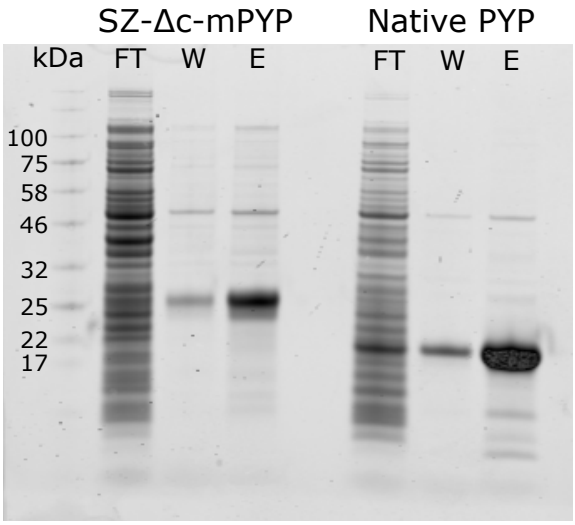


Figure 2.5: Protein gel for expressed proteins. The flow through, wash, and elution solutions from Ni-NTA purification are shown for both proteins expressed. Both SZ-Δc-mPYP (21.5 kDa) and native PYP (14.7 kDa) ran slowly on this gel, though both proteins show approximately the correct relative weights. Under the elution conditions (250 mM imidizole), some amount of other protein was visible; while the lighter bands were likely separated via dialysis, the heavier bands remained in the purified protein product as contaminants.

After dialysis and lyophilization, both proteins were redissolved in Tris buffer for fluorescence and mass spectrometry experiments. Both proteins were also redissolved in phosphate buffer for circular dichroism experiments (10 mM potassium phosphate, 50 mM sodium sulfate) , as the standard 50 mM Tris + 200 mM NaCl buffer absorbs below 200 nm, a crucial region of the UV spectrum for protein structure resolution [21].

Modification

Both proteins were modified by addition of a 10 fold molar excess of p-coumaric anhydride. However, the relative insolubility of these proteins presented challenges in confirming that protein modification had occurred. While native PYP was soluble to around 0.1 mg/mL, the designed SZ-Δc-mPYP showed an even lower solubility around 0.05 mg/mL or lower (these measurements were performed on a NanoDrop, which means that this final measurement is near the lower detection end and thus noisy).

The relative insolubility of these proteins presented challenges in confirming that protein modification with the anhydride had occurred. Absorption measurements for the unmodified and modified conditions of SZ-Δc-mPYP were inconclusive. While we would expect an increase in absorption at the excitation peak if the mutant was photoactive, we instead see a small drop in absorption around 350nm between the unmodified and modified cases as seen in Figure 2.6. Because of the overall solubility issue, it is unclear if this indicates a failure to integrate the chromophore.

On the other hand, the purified native PYP protein does show an emergence of an absorption peak under the modified condition. While the data is still near the lower end of the detection range, a clear peak around 450nm is visible. As the literature excitation maximum for PYP is 446nm [22], this suggests that reconstitution of photoactive PYP was successful.

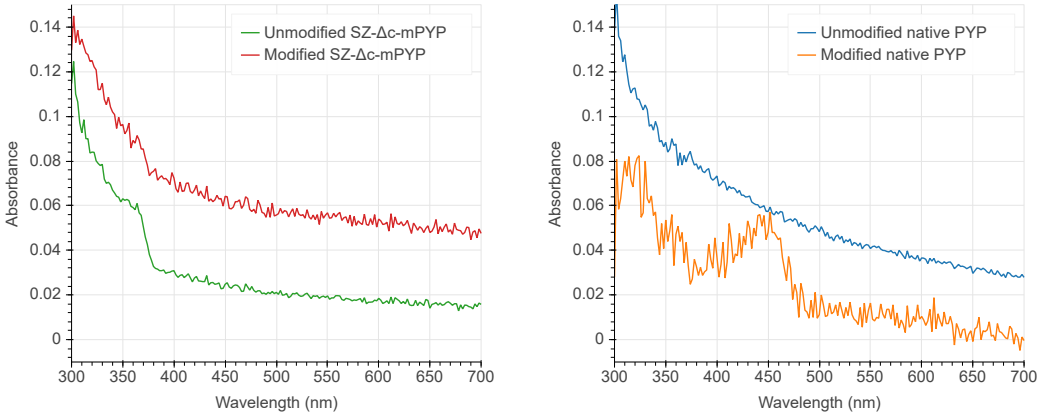


Figure 2.6: Absorbance data for both native PYP and SZ-Δc-mPYP. While the SZ-Δc-mPYP data does not show any visible peaks, the native PYP absorbance data shows a peak around 446nm, the peak of the excitation spectra of PYP. Absorption measurements have been blank-subtracted and been linearly scaled based on differences in loading concentration.

Successful modification of at least the native PYP structure was further confirmed via mass spectrometry. While all four conditions (unmodified and modified plus native PYP versus SZ-Δc-mPYP) were analyzed via MALDI and LC-MS, only the native modified PYP sample gave measurable mass spectra. As seen in Figure 2.7 this mass spectrum did show a strong mass peak one amu smaller than the predicted mass of native PYP reconstituted with its chromophore, indicating that the addition of anhydride lead to successful modification of the native protein.

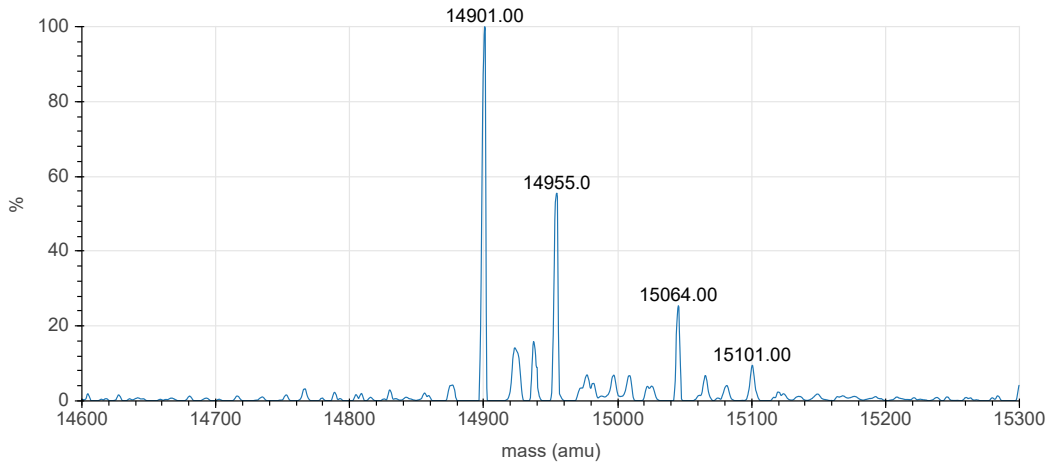


Figure 2.7: MALDI mass spectrum for modified native PYP. The largest peak on the mass spectrum is one amu lower than the expected molecular weight of thiophenyl coumarate-modified native PYP, 14.902 kDa. Unmodified native PYP has a molecular weight of 14.754 kDa.

Circular dichroism measurements

Circular dichroism data is typically normalized against the concentration of the protein. Due to the low concentration of these samples and the inaccuracies of the NanoDrop at these low concentrations, an accurate concentration was not available for normalization purposes.

Two sets of spectra were collected. In the first, CD spectra of a blank cell, unmodified SZ-Δc-mPYP, and unmodified native PYP was taken. In the second, CD spectra of a blank cell (now with the same concentration of anhydride added), modified SZ-Δc-mPYP, and modified native PYP was taken.

In the unmodified spectra, seen after blank-subtraction in Figure 2.8, protein secondary structure is visible. Because the data were not normalized by protein concentration, the magnitude of the ellipticity signal in the SZ-Δc-mPYP

unmodified case is smaller because of its reduced solubility. However, both native PYP and SZ-Δc-mPYP start with a region of positive ellipticity below 200nm and show a dip to negative ellipticity at wavelengths above 200nm—these are both characteristic of the behavior of proteins with well defined α helices and antiparallel β sheets. Importantly, the unmodified proteins do not show negative ellipticity around 195nm which is characteristic of disordered, unfolded proteins [21].

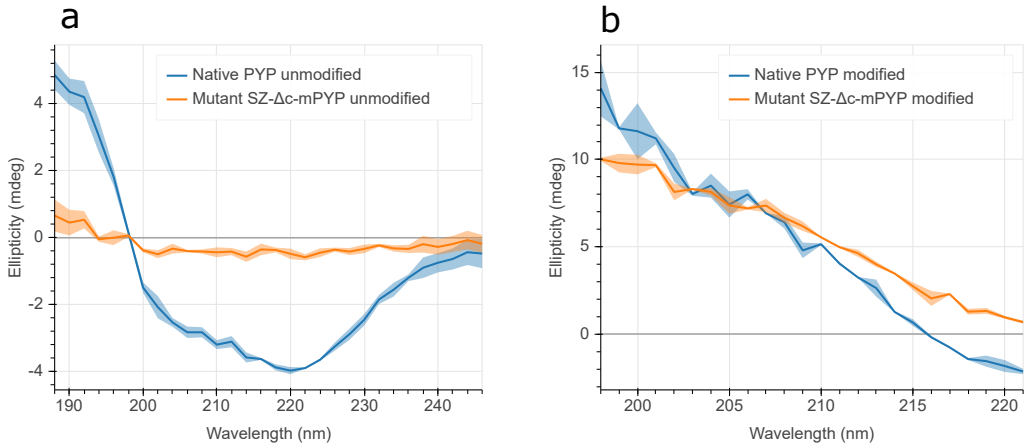


Figure 2.8: Circular dichroism spectra of native PYP and our mutant. Circular dichroism data is shown after blank-subtraction. Plotted data is the average of three measurements; the shaded region visually shows ±one standard deviation. **a:** The absolute ellipticity of the unmodified proteins shows the presence of normal protein secondary structure. This behavior agrees with reported spectra for similar PYP mutants in the literature [10, 11] **b:** The absolute ellipticity of the modified proteins shows a large positive ellipticity, possibly indicating additional sample purification is required after modification.

In the modified spectra, even after blank subtraction, the data remains inconclusive. The large positive ellipticity seen is inconsistent with the CD spectra of reconstituted PYP mutants reported in the literature [11]. This likely indicates that the samples modified with anhydride were not purified properly for CD analysis.

2.5 Cellular aggregation

In light of the solubility issues of the reconstituted proteins, cellular aggregation experiments were performed using the surface displayable constructs. The four constructs created in this work as described in Figure 2.3b were used with constructs that surface displayed SynZip17 or SynZip18, previ-

ously created by the Tirrell lab [5]. The SynZip17 construct were previously transformed into *E. coli* constitutively expressing mWasabi, a green fluorescent protein, whereas the SynZip18 construct was previously transformed into *E. coli* constitutively expressing mCherry, a red fluorescent protein. The four PYP-derived constructs were transformed separately into mWasabi and mCherry expressing *E. coli* such that in the experimental aggregation conditions the different bacterial strains could be distinguishable through confocal microscopy.

In microscopy images for conditions with one of the designed constructs, the cells expressing our designed constructs have been false-colored to be magenta and cells expressing SynZip17 or SynZip18 have been false-colored to be green. All microscopy images are presented as maximum-intensity projections of z-stacks.

Nonspecific interactions of native PYP constructs and of capless Δc-mPYP-AT

In order to connect aggregation induced by our SZ-Δc-mPYP to the design of the SZ cap region, we needed to demonstrate that surface displayed native PYP or Δc-mPYP could not induce either self-aggregation or partner-specific aggregation. In order to show this, control samples of single strains of bacteria were first used to check for self-aggregation and strains surface displaying native PYP, capless native PYP, and Δc-mPYP were mixed with strains either expressing SynZip17 or SynZip18 to check for cross-aggregation behavior.

Of the single-strain controls, only the Δc-mPYP sample showed large amounts of self-aggregation. Other proteins, including our target SZ-Δc-mPYP-AT, did not show self-aggregation. However, for Δc-mPYP, seen in Figure 2.9, large self-aggregates formed rapidly.

We attribute this aggregation behavior to the highly hydrophobic core region that normally interacts with the cap region of the protein. By removing the designed SZ cap, the core hydrophobic β-sheets can likely drive nonspecific aggregation through hydrophobic packing.

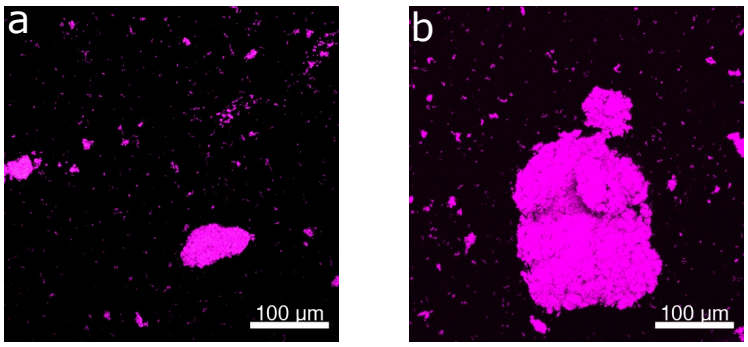


Figure 2.9: Self-aggregation of bacteria surface displaying ∆c-mPYP, the capless mutant protein. a: Microscopy image of ∆c-mPYP-AT two hours after induction. **b:** Microscopy image of ∆c-mPYP-AT four hours after induction.

A similar self-aggregation behavior is not seen in the strains surface displaying native PYP or capless native PYP, though some self-aggregates of bacteria surface displaying SynZip18 are evident. Additionally, as seen in Figure 2.10, some amount of interaction between these control strains and SynZip17/18 strains is visible, though no large cross-aggregates are visible. Interestingly, the self-aggregation properties of ∆c-mPYP-AT is modulated by the presence of SynZip18 expressing bacteria. When ∆c-mPYP-AT is mixed with cells surface displaying SynZip17, similarly sized self-aggregates are visible.

In contrast, the self-aggregates visible in the ∆c-mPYP-AT/SynZip18 case are much smaller. One small aggregate also appears to include intercalated cells expressing SynZip18. One possible explanation would be that intercellullar interactions between SynZip18 and ∆c-mPYP occur transiently but specifically, inhibiting the nonspecific hydrophobic interactions. This may be possible because of the design method used to optimize the SynZip cap binding site in the SZ-∆c-mPYP protein. The core binding pocket region was designed to selectively interact with a cap region similar in sequence to SynZip17, meaning that it might share some rudimentary interactions with SynZip18, a protein that also selectively interacts with SynZip17. If SynZip18 is interacting with the core region of ∆c-mPYP, the nonspecific hydrophobic interactions likely driving self-aggregation could be disrupted. To properly test this hypothesis, soluble SynZip17/SynZip18 could be added to examine the effects on self-aggregation of the capless mutant.

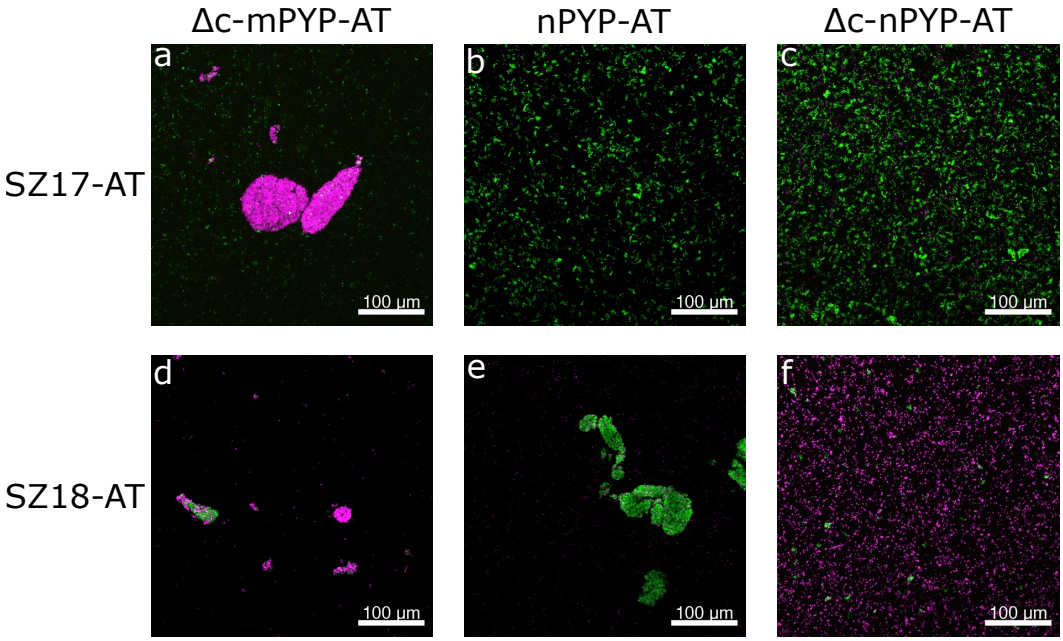


Figure 2.10: Cross-interactions between strains surface displaying SynZip constructs and strains surface displaying control constructs. SynZip strains are shown in green, control strains are shown in magenta. Samples were prepared two hours after induction and one hour after cultures were mixed.**a:** In the presence of SynZip17 expressing cells, the capless mutant still forms small self-aggregates as seen in Figure 2.9. **d:** In the presence of SynZip18, self-aggregation of Δc-mPYP appears to be inhibited, with the formation of possible small mixed aggregates. **e:** Some amount of SynZip18 self-aggregation is visible. **b, c, f:** No appreciable aggregation occurred.

Selective aggregation of the designed SZ-Δc-mPYP protein.

With these controls showing no significant cross-aggregation behavior, we now examine the behavior of surface displayed SZ-Δc-mPYP. As seen in Figure 2.11, the SZ-Δc-mPYP forms large, $\approx 100\mu m$ aggregates when mixed with a strain expressing SynZip18, about the same size as the aggregates formed in a control mixture of SynZip17/SynZip18 strains. Similarly sized aggregates are not formed when SZ-Δc-mPYP-AT is mixed with a strain expressing SynZip17. The aggregates formed by SZ-Δc-mPYP/SynZip17 also appear to be well intercalated, indicating that cross aggregation between these two is in fact driving aggregation.

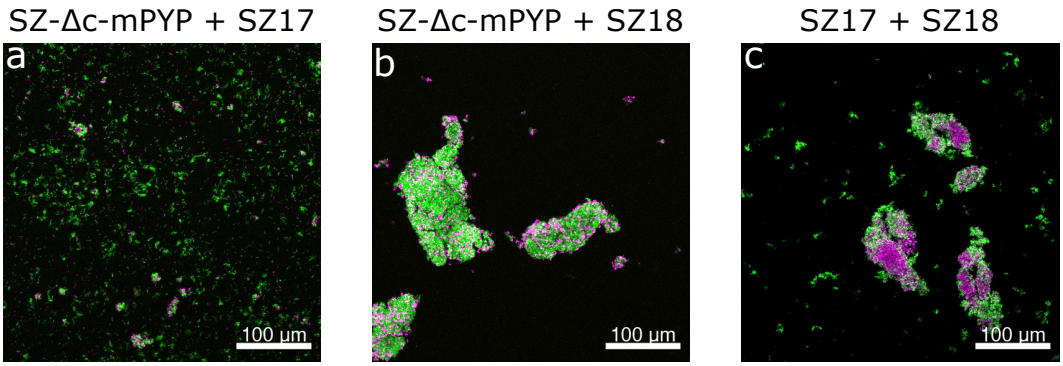


Figure 2.11: Selective aggregation properties of designed SZ17-PYP. a: Only some small (around 10 micron) aggregates are seen when strains surface displaying SZ-Δc-mPYP and SynZip17 are mixed. **b:** Large aggregates are visible when strains surface displaying SZ-Δc-mPYP and SynZip18 are mixed, indicating that the mutant SZ-PYP protein is capable of selectively aggregating with SZ18 but not with SZ17. **c:** Similarly sized aggregates are visible in this control case, where strains surface displaying SynZip17 and SynZip18 were mixed.

Photoactivity of protein constructs

Finally, these aggregation experiments were also performed both with and without the presence of added thiophenyl coumarate. Successful integration of the chromophore into the proteins was examined by using a separate acquisition channel with the laser excitation frequency set to excite PYP and the emission filter frequencies set to encompass PYPs emission spectra. Outside of low-intensity background likely caused by a slight overlap in PYP's and mWasabi's emission spectra, no yellow fluorescence is visible in conditions where thioester was not added. However, all conditions where thioester were added showed small, high-intensity regions of yellow fluorescence, as seen in the control samples in Figure 2.12.

While some cases, such as Figure 2.12c and Figure 2.12d, appeared to show localization of these yellow spots inside small self-aggregates, other cases appear to show fluorescence outside of cells. Yellow spots were also visible in the strains only expressing SynZip17 and SynZip18. This suggests that some other process other than proper integration of the chromophore into PYP-derived proteins is causing this fluorescence. As the photoactivity of PYP comes from the trans-cis isomerization of the attached thiophenyl coumarate group, it may be possible that in addition to binding to the chromophore site in PYP-derived proteins, the thioester may be non-specifically modifying

other free cysteine residues. If such a bound thiophenyl coumarate is capable of causing weak fluorescence without the full chromophore binding pocket, we might expect local areas of fluorescing thiophenyl coumarate.

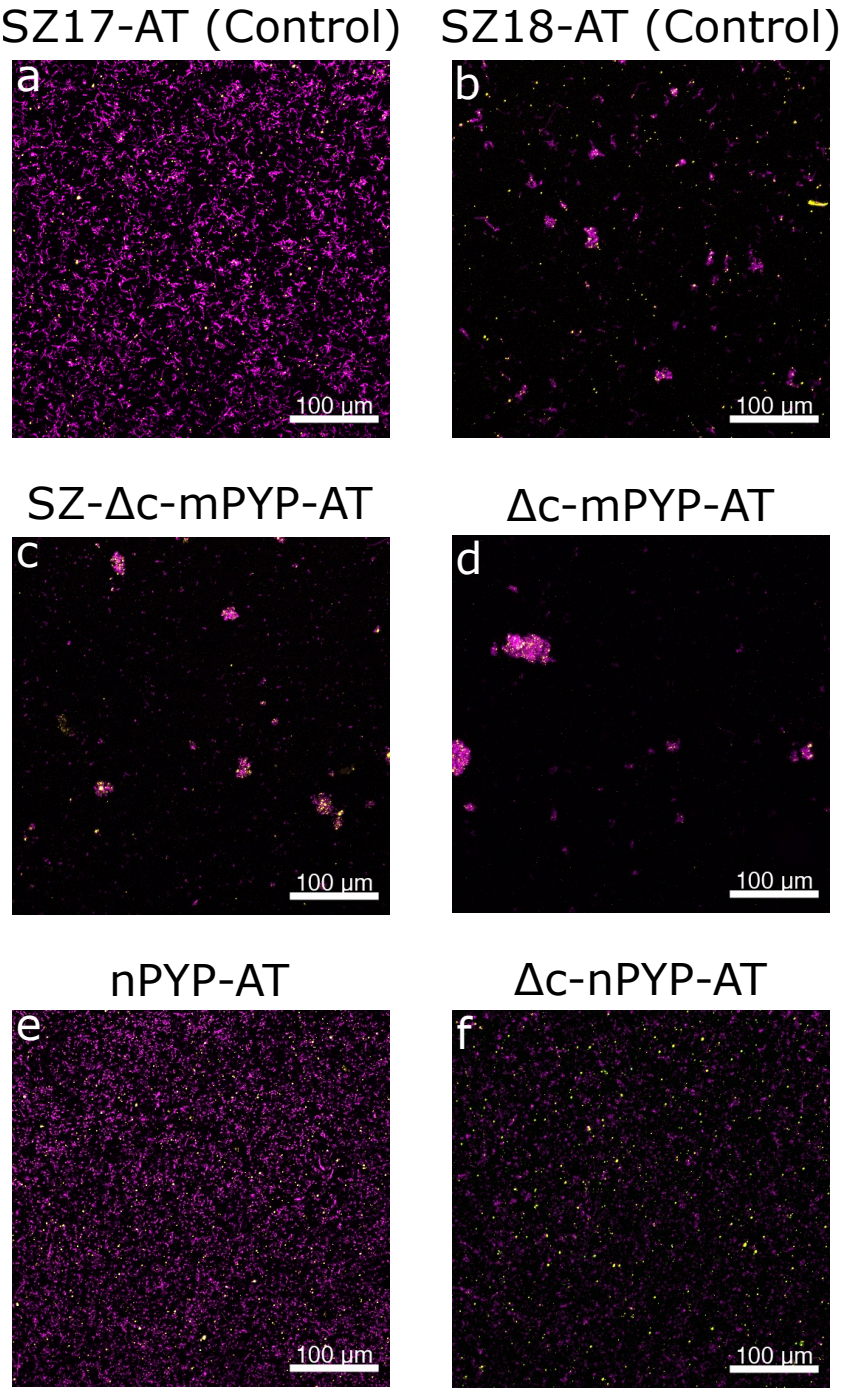


Figure 2.12: Putative thioester modified samples. The constitutively expressed fluorescent protein used to image bacterial cells is shown in magenta, whereas the activity at PYPs emission/excitation wavelengths is shown in yellow.

Chapter 3

DISCUSSION AND FUTURE WORK

3.1 Computational design effectiveness

The original computational design protocol was designed to create a mutant cap with behavior similar to SynZip17 while simultaneously redesigning the core region to selectively sequester this cap in the light state. While we were unable to directly confirm the photoactivity of this mutant protein, the designed mutant showed selective aggregation behavior against SynZip18, confirming that both the design method is capable of building a largely modified cap region with the target behavior. This selective aggregation behavior seen experimentally agrees with the coiled coil interaction prediction model [15] used here to validate the design prior to our expression experiments.

The resulting designed protein also appeared to be much more hydrophobic than the native PYP structure, itself only sparingly soluble. Some of this additional hydrophobicity is expected. As we are designing the cap region to include the residues from the hydrophobic face of SynZip17, optimization of the protein folding energy means that we also induce the core "cap binding pocket" region to become more hydrophobic to create favorable hydrophobic packing. As seen in the overall low solubility of the designed mutants and strong nonspecific hydrophobic interactions likely driving self-aggregation of the capless mutant protein, strongly optimizing for folding energy can have deleterious effects when the resulting protein is expressed. By constraining optimization away from highly hydrophobic solutions, favorable solutions that balance SynZip-like behavior with lower overall hydrophobicity may be possible.

Finally, our computational modeling method predicted that one of the seven key residues interacting with the chromophore should be mutated. While the low expression of the SZ-Δc-nPYP construct meant that the photoactivity of the mutated PYP core versus the native PYP core could not be compared, Rosetta does not perform any modeling of the active behavior of ligands. This means that the computational design may have destabilized the chromophore binding site, possibly affecting the photoactivity of the mutant. Explicit design

constraints on residues interacting with the chromophore could solve this problem.

3.2 Validation of photoswitchable behavior

Due to time constraints and the solubility issues inherent with performing purified protein tests on the designed SZ-Δc-mPYP, photoswitchable behavior was not fully demonstrated on the mutant protein. However, we demonstrated that native PYP was modified via addition of the anhydride and that after this modification, the native PYP showed a new absorption peak near the reported excitation max of PYP.

However, modification of SZ-Δc-mPYP could not be seen via fluorescence or mass spectrometry. The lack of photoactivity in the mutant may be caused either by failure to incorporate the chromophore or an insufficient concentration of photoactive SZ-Δc-mPYP due to the solubility issues. Neither different resuspension buffers nor small amounts of surfactants such as Triton X-100 were able to significantly increase the solubility of SZ-Δc-mPYP, making direct fluorescent measurements difficult.

In the literature, reconstitution of photoactive PYP mutants is performed in non-denaturing resuspension buffers [11, 18]. The extraordinarily limited solubility of our mutant in non-denaturing buffers thus made mass spectrometry difficult; without sufficient soluble protein, mass spectra could not be collected on the putative modified protein.

If the thioester modification is stable under some set of denaturing conditions, the solubility limitation could possibly be avoided by modifying SZ-Δc-mPYP in non-denaturing buffers, followed by desalting, concentration, and resuspension in a denaturing buffer. This would allow mass spectrometry to proceed on a higher-concentration sample.

Additionally, use of a more accurate protein concentration assay, such as BCA assay, instead of NanoDrop absorbance measurements would aid in optimization of reconstitution conditions and would provide valuable normalization coefficients for the fluorescence and CD measurements. Such an assay was not performed as part of this work due to time constraints and lack of readily-available reagents.

3.3 Optimization of photoswitchable activity

If the solubility issues could be solved and the resulting protein does not show the desired fold-change in aggregation activity between the light and dark states, further optimization of this system could be performed using computationally-informed directed evolution. In the process of selecting SZ-Δc-mPYP, hundreds of predicted mutants were generated by Rosetta. By examining which residues in this computational library affected protein stability,key residues could be selected for site-saturation mutagenesis. Then, using a FRET assay similar to that initially used to characterize the SynZip library, library members could be screened via plate reader for mutants that showed a large fold-scale difference in association between the light and dark states [8]. This directed evolution procedure could identify a derived mutant with favorable photoswitchable properties.

An alternative to directed evolution could be a wider screen of computationally-predicted mutants. However, Rosetta design computations are very time-intensive; prediction and decoy validation of the single mutant tested in this work took several hundred CPU hours. Even though this prediction was performed on scalable cloud computing resources, a FRET-based directed evolution experiment would be able to set up and screen a much larger effective library than the alternative—cloning and testing individual computationally-predicted mutants.

3.4 Creation of a general-purpose photoswitchable SynZip aggregation system

We have demonstrated the ability to computationally design mutant proteins that show selective coiled-coil based aggregation behaviors. This means that design of more complicated bacterial aggregates could be feasible computationally. Beyond the straightforward bioreactor startup and shutdown applications of light-drive aggregation and disaggregation, the system could be extended to other designed mutants, based on the structures of the versatile SynZip library. One of the advantages of the SynZip library is the number of orthogonal association pairs, as several SynZip library members can be selected with desired amounts of cross- or self-association between each member. The relative flexibility of these library members to computational redesign shown here indicates that creation of multiple disparate photoswitchable SynZip variants may be possible.

If full photoswitchability can be demonstrated, control of bacterial aggregates on the timescale of seconds or minutes would be possible.

Even using the quorum sensing aggregation circuit demonstrated in Kozlowski *et al.* [5], aggregate morphology and behavior is still limited by the timescale of protein synthesis and the relative speed of inducible promoters. If precise temporal control is combined with the association networks underlying the SynZip library, we could form complex bacterial aggregates, such as a multi-layered core-shell structure, with light. Such a system would otherwise require combining separately cultured bacteria at specific timepoints or the addition of large amounts of exogenous inducer

Finally, if the timescale of the light/dark transition remains around the several second transition timescale of native PYP, aggregate size and composition could be dynamically controlled using modulated light levels.

Chapter 4

METHODS

4.1 Computational modeling

As we are not performing this design *de novo*, it is convenient to start design using a pre-minimized structure. Thus, the crystal structure for native PYP was relaxed using Rosetta to generate a "modeling-ready" version of the structure that already exists at a local minimum in Rosetta's energy landscape. Only very small (generally less than an angstrom) changes to the structure were made by this relaxation step.

Four RosettaScripts, included in Appendix B, were created in order to carry out the computational modeling steps. These four carried out 1) the N-terminal cap design, 2) the light-state binding design, 3) the dark-state validation decoy generation, and 4) the light-state validation decoy generation. All sidechain repacking steps sampled from a conformational space expanded from the Rosetta defaults by one standard deviation.

The N-terminal cap design script uses four consecutive movers. In the first mover, FastDesign was used to fully redesign (e.g. allow residue identities in the N-terminal cap to change, allow the backbone to move, and allow the sidechain rotamers to move) the N-terminal cap region, subject to the constraint that cap positions corresponding to the a/d/e/g heptad locations match the SZ17 residue, in addition to allowing the binding site region sidechain rotamers to be repacked. The base protein structure used was a crystal structure of native PYP (PDB 1NWZ). Then, the next mover, PackRotomersMover, was used to redesign the binding site, allowing binding site residues to change identities if possible. This step is responsible for fixing problems introduced by the cap redesign, and was allowed to take actions like mutating newly exposed hydrophobic residues. Finally, a round of deterministic minimization and stochastic relaxation was done prior to energy scoring. 120 predicted structures (45 minutes of CPU time, 2GB of memory per structure) were generated in this way and the protein with the highest predicted stability was selected.

Then, this protein was used to perform "light-state design". The light-state

design first remodeled the protein sequence generated by the N-terminal cap design step using PDB structures of PYP in the light state and SynZip5. This remodel effectively "threads" the generated protein sequence onto a combination of these two structures, creating a hybrid protein structure. Then, the fold tree, a core Rosetta concept that determines how rigid body motions propagate through the rest of the protein structure, was set up with its root at the linker region between the N-terminal cap and the rest of PYP. After these threading and fold tree manipulations were treated, the backbone residue angles for the "linker" region were perturbed, effectively sampling the possible angular orientations that the SynZip cap could take on. This perturbation was followed by a deterministic minimization, followed by a binding site redesign step that repacked rotamers for both the N-terminal SynZip cap and the associated binding site, but only allowed residue identities to change in the binding site. This design step was followed by a stochastic relaxation step. 300 predicted structures (15 minutes of CPU time, 3GB of memory per structure) were generated by this RosettaScript, and the protein with the highest predicted stability was selected and designated as the SZ17-PYP mutant reported here.

Decoy validation was done for both the light and dark state. The selected mutant was re-threaded and had its fold tree set up in a similar way to the light state design script. Then, the backbone angles of the linker region were perturbed, and the structures were subjected to deterministic minimization and stochastic relaxation to explore the energy landscape for our target protein. Around two thousand structures(5 minutes of CPU time, 1.5GB of memory per structure) for both the light and dark state were generated by these Rosetta scripts.

These computations took several hundred CPU hours. In order to quickly run these design steps, these calculations were carried out on Google Cloud Platform, using custom scripts designed to launch and manage the data generated by parallel Rosetta runs on several virtual machines.

Protein decoy clustering and analysis

Protein decoys were analyzed chiefly through clustering and manual examination of the resulting stable structures. Clustering was performed using a custom implementation of a modified Lloyd’s algorithm, using a full-atom

RMS kernel. Python code for this clustering is included in Appendix C. In short, Lloyd's algorithm clusters objects according to pairwise distances between them. As a clustering algorithm that locates the most optimal clusters is NP-hard, Lloyd's algorithm stochastically finds "good" clusters by alternating between a phase where cluster centers are assigned and one where objects are assigned to the closest structure. In this case, it means that decoys were randomly assigned as "centers". Then, proteins were assigned to "clusters" based on the center to which the mean RMS was the smallest. Then, for each cluster, 40 cluster members were selected and evaluated if they were a better "center" than the randomly selected center (e.g. if using that cluster member as the "center" resulted in a smaller mean RMS across all cluster members). This procedure was then repeated with the newly selected centers until convergence. An example cluster of decoys predicted by this modified Lloyd's algorithm is shown in Figure 4.1.

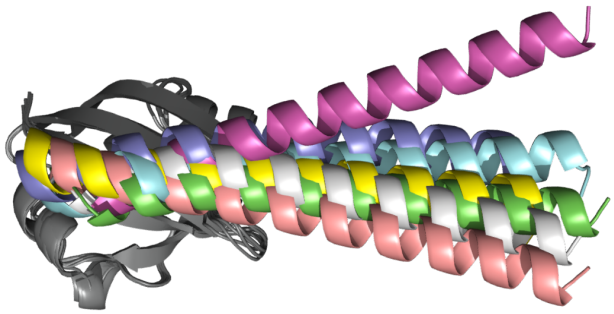


Figure 4.1: A set of clustered decoys identified through Lloyd's algorithm. These proteins were "projected" into a high-dimensional space where distance between proteins is measured through the average RMS atom distance, then clustered with Lloyd's algorithm.

4.2 Molecular cloning

Molecular cloning was performed using standard restriction enzyme and Gibson assembly techniques. Constructs were created to be inserted into a modified version of the pQE80 cloning vector (containing an ampicillin resistance cassette and an IPTG-inducible promoter) that had an extra XhoI cloning site removed. Due to the difficulty of introducing a restriction site into the open reading frame of SZ17-PYP, the construct SZ-Δc-nPYP was constructed via Gibson assembly, with primers used to add flanking overlap regions. All constructs were transformed into competent Mach1 *E. coli* cells

and selected using LB-carbenicillin plates (100 ug/mL). Plasmids were prepared with the standard Qiagen miniprep kit; PCR reactions were performed with Q5 polymerase (NEB); digestion reactions were performed with HindIII-HF, EcoRI-HF, DpnI, and XhoI (NEB); and Gibson assembly was performed with Gibson Master Mix (NEB). Gblock gene constructs, cloning primers, and sequencing verification primers were ordered through IDT.

Genetic constructs for His-tagged SZ17, SZ18, and their surface-displayable versions SZ17-Autotransporter (AT) and SZ18-AT had previously been constructed and were provided for this work by the Tirrell lab. The four additional surface-displayable genetic constructs were prepared under an IPTG-inducible promoter via standard molecular cloning techniques. All of these additional proteins—see Figure 2.3 for an illustration of the constructs and Appendix A for sequences—were successfully cloned into the pQE80 vector. While not explicitly listed here, all protein constructs include a 6x His tag for purification and surface displayable constructs include the N-terminal secretion tag pelB.

4.3 Protein expression and purification

From a glycerol stock, a 5mL culture of the relevant bacterial strains was grown for four hours. This culture was used to inoculate 50mL cultures in LB + carbenicillin (100 ug/mL), which were allowed to grow overnight. 25 mL of these cultures were used to inoculate one-liter cultures in Terrific Broth (TB) media, containing the same concentration of antibiotic. These cultures were allowed to grow for two hours at 37°C, after which protein expression was induced with 1 mM IPTG. After four hours, cells were collected via centrifugation. Standard Ni-NTA His tag purification procedures, as described in the QIAExpressionist, will be used to purify the cell lysate [23], except for the replacement of Buffer C and D with Buffer “I50” and “I250”. Buffer I50 is pH8 Buffer B plus 50 mM imidazole and Buffer I250 is pH8 Buffer B plus 250 mM imidazole.

4.4 p-coumaric anhydride synthesis

252.6mg of p-coumaric acid was dissolved in 21 mL of DMF. 492.8mg of EDC (0.5 equivalents) was added to the reaction mixture. The reaction was allowed to stir on ice overnight. The resulting mixture was extracted with ethyl acetate and water (500 mL water, 150 mL brine, 400 mL ethyl acetate).

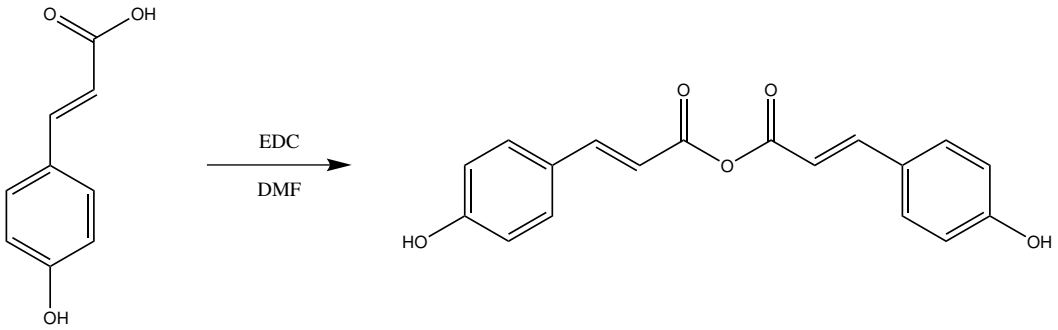


Figure 4.2: Synthesis of p-coumaric anhydride from p-coumaric acid. In the presence of 1-Ethyl-3-(3-dimethylaminopropyl)carbodiimide (EDC) as an activator, p-coumaric anhydride is formed through removal of a urea byproduct.

After drying with magnesium sulfate, the ethyl acetate layer was removed via rotovap. After further vacuum drying, 225.4 mg (47.7% yield) was obtained. A proton NMR in methanol-d_4 was taken both of this crude product and of the starting material, p-coumaric acid. Using peaks assigned from the clean starting material NMR, as seen in Figure 4.3, the shifted peaks corresponding to the anhydride product were identified.

The initial crude product NMR showed many impurities, so the product was redissolved in ethyl acetate and was extracted using a water/bicarbonate base solution, where we extract the relatively nonpolar anhydride in the ethyl acetate layer. After additional rotary evaporation and vacuum drying, 80mg of product was isolated. Another NMR was taken of the resulting product. The doublet at 6.26ppm, assigned to the vinylic proton closest to the carboyxlic acid group, was chosen as a diagnostic group because of the relative sparsity of signals in that region. We also chose this doublet as the diagnostic group because we expect it to be the proton with the largest change in chemical shift when comparing the acid and anhydride forms because of its proximity to the carboxylic acid. As seen in Table 4.1 and Figure 4.4, the proton integral of these peaks indicate that even after base extraction, our final mixture is at a molar ratio of around 2 acid : 1 anhydride. While some of the acid may be created from hydrolysis from the methanol used as the NMR solvent, this likely indicates that purification by extraction was not sufficient to remove all of the p-coumaric acid starting material.

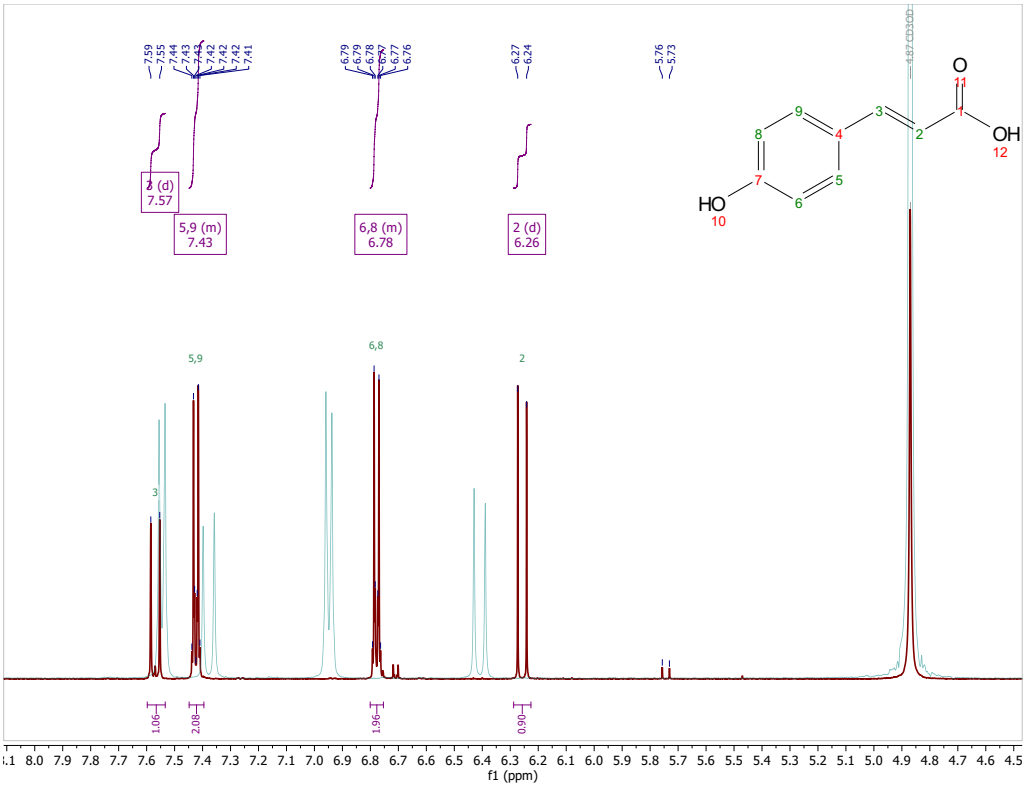


Figure 4.3: Proton NMR spectrum of p-coumaric acid. Several well-defined peaks are visible in this spectrum, making assignment of peaks to the protons in the structure straightforward. Literature spectrum from the Biological Magnetic Resonance Data Bank shown in cyan (bmse000150). The spectra were aligned based on the methanol peak. Around a 0.15ppm shift is seen between the spectra under this alignment.

NMR peak	Assignment	Proton integral	Molar equivalents
6.26 (d)	Vinylic proton in the acid	1.00	1.00
6.43 (d)	Vinylic proton in the anhydride	.83	0.415
6.78 (m)	Aromatic protons in the acid	1.93	1.93
7.42 (m)	Aromatic protons in the acid	1.96	1.96
6.82 (m)	Aromatic protons in the anhydride	3.79	1.89
7.53 (m)	Aromatic protons in the anhydride	3.55	1.77

Table 4.1: Summary assignments from proton NMR on extracted p-coumaric anhydride product. Because the aromatic multiplets are heavily overlapping, we rely on the diagnostic doublets to infer the molar ratio of acid to anhydride.

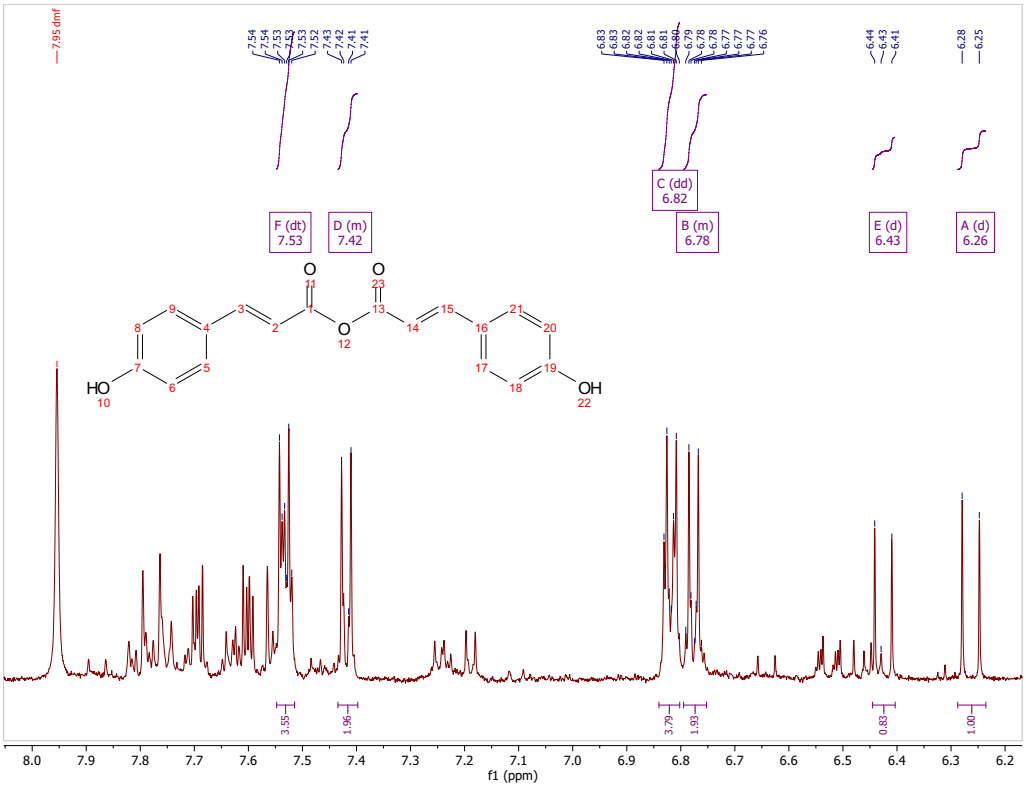


Figure 4.4: Proton NMR spectrum of the p-coumaric anhydride product. The aromatic peak region of the product is heavily overlapping, though several diagnostic pairs of peaks, such as peaks A and E, were identified as coming from p-coumaric acid and p-coumaric anhydride.

4.5 Thiophenyl coumarate synthesis from anhydride

40 mg of p-coumaric anhydride previously synthesized was dissolved in 5 mL of THF. 7.04mg of 4-dimethylaminopyridine (0.4 molar equivalents) and 22 microliters of thiophenol (23.8 mg, 1.5 molar equivalents) were added. The reaction was allowed to stir at room temperature overnight, followed by a basic ethyl acetate/water extraction. 10 mg of product was recovered, and a proton NMR of the product was done. However, the resulting product was too dilute for proper peak analysis. Instead, the synthesis of thiophenyl coumarate was performed using a direct route from p-coumaric acid.

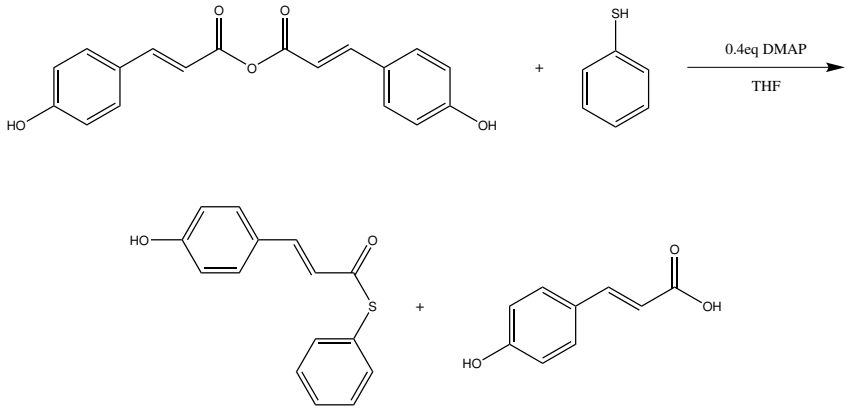


Figure 4.5: Synthesis of thiophenyl coumarate from p-coumaric anhydride. In the presence of DMAP, a catalytic nucleophilic base, p-coumaric anhydride reacts with thiophenol to form equal amounts of the thiophenyl coumarate and p-coumaric acid.

4.6 Thiophenyl coumarate synthesis from carboxylic acid

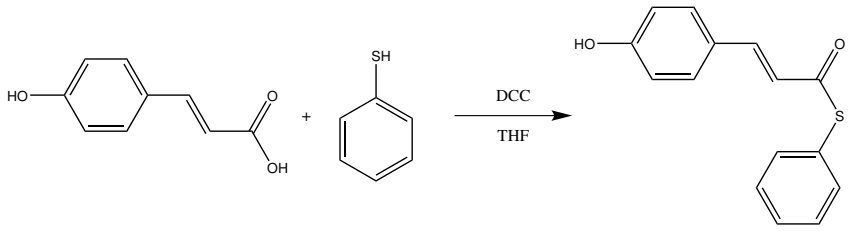


Figure 4.6: Synthesis of thiophenyl coumarate from p-coumaric acid. In the presence of *N,N*'-Dicyclohexylcarbodiimide (DCC), a dehydrating agent, thiophenyl coumarate can be directly formed from p-coumaric acid and thiophenol.

200mg of p-coumaric acid was dissolved in 20 mL of THF. 301.45mg of DCC (1.2 molar equivalents) and 149.58 microliters (160.95mg, 1.2 molar equivalents) of thiophenol were added to the mixture. The reaction was stirred at

room temperature overnight, then worked up via basic water/ethyl acetate extraction. The resulting product was separated using an automated chromatography column (Biotage), using a solvent gradient from 98 DCM:2 MeOH to 80 DCM:20 MeOH. The fraction that eluted around 85 DCM:15 MeOH was collected and dried via rotary evaporation and vacuum drying. 300mg of product were recovered. Due to the similarity between the p-coumaric acid and the p-coumaric thiopheonl and the increased density of peaks in the aromatic region, it was difficult to determine a molar ratio of the final extracted product (see Figure 4.7). The vinylic proton peak closest to the aromatic group (proton 3 in Figure 4.3) was chosen as an additional diagnostic peak. Using the peaks at 7.50 and 7.60ppm, the isolated product had a molar ratio of 2.5 thiophenyl coumarate : 1 p-coumaric acid. Using the peaks at around 6.55ppm and 6.51ppm, the isolated product had a molar ratio of 1 thiophenyl coumarate : 1 p-coumaric acid.

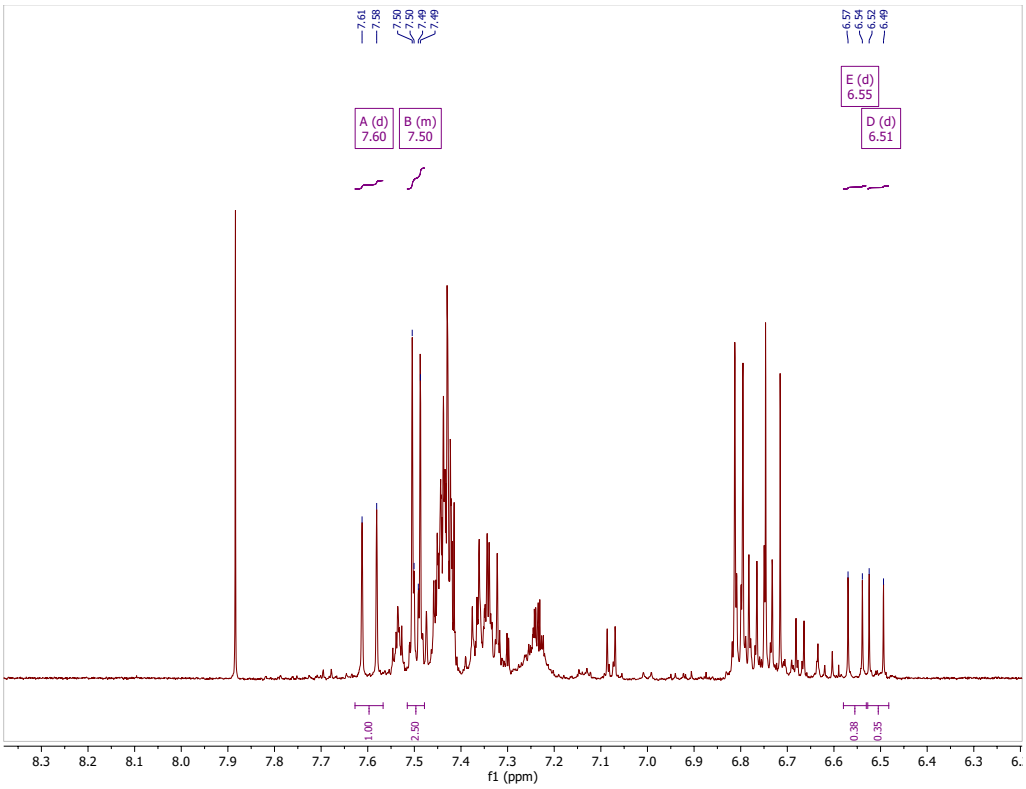


Figure 4.7: Proton NMR of the thiophenyl coumarate product. Due to the many overlapping peaks, only two diagnostic peaks corresponding to the vinylic protons were assigned. These peaks showed a molar ratio of thiophenyl coumarate to p-coumaric acid between 1:1 and 2.5:1.

4.7 Modification of purified proteins

0.3mg of purified native PYP and SZ-Δc-mPYP were dissolved in two different buffers. One buffer was a Tris buffer (50 mM Tris, 200 mM NaCl, pH 8) and the other was a phosphate buffer (10 mM potassium phosphate, 50 mM sodium sulfate, pH 8). Then, p-coumaric anhydride dissolved in DMF was added to the buffers until there was a ten-fold molar excess of the anhydride. This was allowed to react for one hour. This mixture was used directly for absorbance and circular dichroism experiments; prior to mass spectrometry, the mixtures were desalted and concentrated.

4.8 Cellular aggregation experiments

From overnight cultures of strains expressing all constructs used in the aggregation experiments (SZ17-AT,mWasabi; SZ18-AT, mCherry; SZ-Δc-mPYP-AT, mWasabi; SZ-Δc-mPYP-AT, mCherry; Δc-mPYP-AT, mWasabi; Δc-mPYP-AT, mCherry; nPYP-AT, mWasabi; nPYP-AT, mCherry; Δc-nPYP-AT,mWasabi; Δc-nPYP-AT,mCherry), cultures were induced at an OD600 of .01 into 5mL of fresh LB media + 100 μg carbenicillin/mL (for plasmid maintenance). Cultures were grown at 37°C and were shaken at 330rpm. After two hours and fifteen minutes (approximate OD600 of 0.6), the cultures were induced with 0.1 mM IPTG and split into 2.5 mL cultures. One of the 2.5mL cultures in each pair had a solution of thiophenyl coumarate added to a final concentration of 25 mg/L. After one hour, sixteen mixed cultures were created (four cultures made from mixing SZ17-AT,mWasabi with each of the experimental mCherry constructs, four cultures from mixing SZ18-AT,mCherry with each of the experimental mWasabi constructs, plus the eight equivalent cultures with added thioester). After an additional hour, 5 μL aliquots were made into slides for confocal microscopy.

BIBLIOGRAPHY

1. Vlamakis, H., Chai, Y., Beauregard, P., Losick, R. & Kolter, R. Sticking together: building a biofilm the Bacillus subtilis way. *Nature Reviews Microbiology* **11,** 157–168. ISSN: 1740-1534 (Mar. 2013).

2. Flemming, H.-C. *et al.* Biofilms: an emergent form of bacterial life. *Nature Reviews Microbiology* **14,** 563–575. ISSN: 1740-1534 (Sept. 2016).

3. Kim, H. J., Du, W. & Ismagilov, R. F. Complex function by design using spatially pre-structured synthetic microbial communities: degradation of pentachlorophenol in the presence of Hg(ii). *Integrative Biology: Quantitative Biosciences from Nano to Macro* **3,** 126–133. ISSN: 1757-9708 (Feb. 2011).

4. Ramsden, J. I. *et al.* Biocatalytic N-Alkylation of Amines Using Either Primary Alcohols or Carboxylic Acids via Reductive Aminase Cascades. *Journal of the American Chemical Society* **141,** 1201–1206. ISSN: 0002-7863 (Jan. 23, 2019).

5. Kozlowski, M., Silverman, B., Johnstone, C. & Tirrell, D. Genetically-programmable assembly of microbial aggregates triggers a quorum sensing response. *Unpublished manuscript.* (2019).

6. Jose, J. Autodisplay: efficient bacterial surface display of recombinant proteins. *Applied Microbiology and Biotechnology* **69,** 607–614. ISSN: 1432-0614 (Feb. 1, 2006).

7. Sun, F., Zhang, W.-B., Mahdavi, A., Arnold, F. H. & Tirrell, D. A. Synthesis of bioactive protein hydrogels by genetically encoded SpyTag-SpyCatcher chemistry. *Proceedings of the National Academy of Sciences* **111,** 11269–11274. ISSN: 0027-8424, 1091-6490 (Aug. 5, 2014).

8. Thompson, K. E., Bashor, C. J., Lim, W. A. & Keating, A. E. SYNZIP Protein Interaction Toolbox: in Vitro and in Vivo Specifications of Heterospecific Coiled-Coil Interaction Domains. *ACS Synthetic Biology* **1,** 118–129. ISSN: 2161-5063 (Apr. 20, 2012).

9. Lungu, O. I. *et al.* Designing Photoswitchable Peptides Using the AsLOV2 Domain. *Chemistry & Biology* **19,** 507–517. ISSN: 1074-5521 (Apr. 20, 2012).

10. Morgan, S.-A. & Woolley, G. A. A photoswitchable DNA-binding protein based on a truncated GCN4-photoactive yellow protein chimera. *Photochemical & Photobiological Sciences* **9,** 1320. ISSN: 1474-905X, 1474-9092 (2010).

11. Ali, A. M. *et al.* Optogenetic Inhibitor of the Transcription Factor CREB. *Chemistry & Biology* **22,** 1531–1539. ISSN: 1074-5521 (Nov. 19, 2015).

12. Mason, J. M. & Arndt, K. M. Coiled Coil Domains: Stability, Specificity, and Biological Implications. *ChemBioChem* **5,** 170–176. ISSN: 1439-7633 (2004).

13. Das, R. & Baker, D. Macromolecular Modeling with Rosetta. *Annual Review of Biochemistry* **77,** 363–382. ISSN: 0066-4154, 1545-4509 (June 2008).

14. Huang, P.-S. *et al.* RosettaRemodel: A Generalized Framework for Flexible Backbone Protein Design. *PLOS ONE* **6,** e24109. ISSN: 1932-6203 (Aug. 31, 2011).

15. Fong, J. H., Keating, A. E. & Singh, M. Predicting specificity in bZIP coiled-coil protein interactions. *Genome Biology* **5,** R11. ISSN: 1474-760X (Jan. 16, 2004).

16. Singh, M. & Kim, P. S. *Towards predicting coiled-coil protein interactions* in. Proceedings of the fifth annual international conference on Computational biology (ACM, Apr. 22, 2001), 279–286. ISBN: 978-1-58113-353-0. doi:10.1145/369133.369238. <http://dl.acm.org/citation.cfm?id=369133.369238> (visited on 06/03/2019).

17. Gromov, E. V., Burghardt, I., Köppel, H. & Cederbaum, L. S. Electronic Structure of the PYP Chromophore in Its Native Protein Environment. *Journal of the American Chemical Society* **129,** 6798–6806. ISSN: 0002-7863 (May 1, 2007).

18. Ui, M. *et al.* Application of photoactive yellow protein as a photoresponsive module for controlling hemolytic activity of staphylococcal alpha-hemolysin. *Chemical Communications* **48,** 4737–4739. ISSN: 1364-548X (Apr. 18, 2012).

19. Bernard, C. *et al.* The Solution Structure of a Transient Photoreceptor Intermediate: delta25 Photoactive Yellow Protein. *Structure* **13,** 953–962. ISSN: 0969-2126 (July 1, 2005).

20. Imamoto, Y., Ito, T., Kataoka, M. & Tokunaga, F. Reconstitution photoactive yellow protein from apoprotein and p-coumaric acid derivatives. *FEBS Letters* **374,** 157–160. ISSN: 00145793 (Oct. 30, 1995).

21. Greenfield, N. J. Using circular dichroism spectra to estimate protein secondary structure. *Nature protocols* **1,** 2876–2890. ISSN: 1754-2189 (2006).

22. Haker, A. *et al.* The Two Photocycles of Photoactive Yellow Protein from Rhodobacter sphaeroides. *Journal of Biological Chemistry* **278,** 8442–8451. ISSN: 0021-9258, 1083-351X (Mar. 7, 2003).

23. Qiagen. *The QIAexpressionist, Fifth edition* June 2003.

Appendix A

CONSTRUCTS

A.1 Rosetta design output: SZ-∆c-mPYP protein sequence

```
NEKEELKSKKAELRDRIFGLKDKRELLKDKIDELRNEISFGVVILDGDGNILKWNGNEGDIT
GRDKNQVGGKNFFKDVAPCTDSPEFYGKFKEGVASGNLSTEFEYTFDYQMTPTKVTVKMHQD
SHGDSYQVHVSRV
```

A.2 SZ-∆c-mPYP gene sequence

```
MKYLLPTAAAGLLLLAAQPAMAMRGSHHHHHHGSVDGSGSGSGSGSGSNEKEELKSKKAELR
DRIFGLKDKRELLKDKIDELRNEISFGVVILDGDGNILKWNGNEGDITGRDKNQVGGKNFFK
DVAPCTDSPEFYGKFKEGVASGNLSTEFEYTFDYQMTPTKVTVKMHQDSHGDSYQVHVSRVG
SGSGSGSGSGSLET

ATGAAATACCTGCTGCCGACCGCTGCTGCTGGTCTGCTGCTCCTCGCTGCCCAGCCGGCGAT
GGCCATGAGAGGATCGCATCACCATCACCATCACGGATCCGTCGACGGTAGCGGTTCCGGGT
CGGGGTCAGGTTCAGGCTCAAACGAAAAAGAAGAACTGAAATCTAAAAAAGCGGAACTGCGT
GACCGTATCTTCGGTCTGAAAGACAAACGTGAACTGCTGAAAGACAAAATCGACGAACTGCG
TAACGAAATCTCTTTCGGTGTTGTTATCCTGGACGGTGACGGTAACATCCTGAAATGGAACG
GTAACGAAGGTGACATCACCGGTCGTGACAAAAACCAGGTTGGTGGTAAAAACTTCTTCAAA
GACGTTGCGCCGTGCACCGACTCTCCGGAATTTTACGGTAAATTCAAAGAAGGTGTTGCGTC
TGGTAACCTGTCTACCGAATTTGAATACACCTTCGACTACCAGATGACCCCGACCAAAGTTA
CCGTTAAAATGCACCAGGACTCTCACGGTGACTCTTACCAGGTTCACGTTTCTCGTGTTGGA
TCTGGAAGTGGCAGCGGATCAGGCAGCGGATCACTCGAGACTTAG
```

A.3 nPYP gene sequence

```
MEHVAFGSEDIENTLAKMDDGQLDGLAFGAIQLDGDGNILQYNAAEGDITGRDPKQVIGKNF
FKDVAPCTDSPEFYGKFKEGVASGNLNTMFEYTFDYQMTPTKVKVHMKKALSGDSYWVFVKR
VGHHHHHH
```

```
ATGGAACATGTGGCGTTTGGCAGCGAAGATATTGAAAACACCCTGGCGAAAATGGATGATGG
CCAGCTGGATGGCCTGGCGTTTGGCGCGATTCAGCTGGATGGCGATGGCAACATTCTGCAGT
ATAACGCGGCGGAAGGCGATATTACCGGCCGCGATCCGAAACAGGTGATTGGCAAAAACTTT
TTTAAAGATGTGGCGCCGTGCACCGATAGCCCGGAATTTTATGGCAAATTTAAAGAAGGCGT
GGCGAGCGGCAACCTGAACACCATGTTTGAATATACCTTTGATTATCAGATGACCCCGACCA
AAGTGAAAGTGCATATGAAAAAAGCGCTGAGCGGCGATAGCTATTGGGTGTTTGTGAAACGC
GTGGGCCATCACCATCACCATCACTAA
```

A.4 SZ-Δc-mPYP-AT gene sequence

```
MKYLLPTAAAGLLLLAAQPAMAMRGSHHHHHHGSVDGSGSGSGSGSGSNEKEELKSKKAELR
DRIFGLKDKRELLKDKIDELRNEISFGVVILDGDGNILKWNGNEGDITGRDKNQVGGKNFFK
DVAPCTDSPEFYGKFKEGVASGNLSTEFEYTFDYQMTPTKVTVKMHQDSHGDSYQVHVSRVG
SGSGSGSGSGSLETPTPGPDLNVDNDLRPEAGSYIANLAAANTMFTTRLHERLGNTYYTDMV
TGEQKQTTMWMRHEGGHNKWRDGSGQLKTQSNRYVLQLGGDVAQWSQNGSDRWHVGVMAGYG
NSDSKTISSRTGYRAKASVNGYSTGLYATWYADDESRNGAYLDSWAQYSWFDNTVKGDDLQS
ESYKSKGFTASLEAGYKHKLAEFNGSQGTRNEWYVQPQAQVTWMGVKADKHRESNGTLVHSN
GDGNVQTRLGVKTWLKSHHKMDDGKSREFQPFVEVNWLHNSKDFSTSMDGVSVTQDGARNIA
EIKTGVEGQLNANLNVWGNVGVQVADRGYNDTSAMVGIKWQF
```

```
ATGAAATACCTGCTGCCGACCGCTGCTGCTGGTCTGCTGCTCCTCGCTGCCCAGCCGGCGAT
GGCCATGAGAGGATCGCATCACCATCACCATCACGGATCCGTCGACGGTAGCGGTTCCGGGT
CGGGGTCAGGTTCAGGCTCAAACGAAAAAGAAGAACTGAAATCTAAAAAAGCGGAACTGCGT
GACCGTATCTTCGGTCTGAAAGACAAACGTGAACTGCTGAAAGACAAAATCGACGAACTGCG
TAACGAAATCTCTTTCGGTGTTGTTATCCTGGACGGTGACGGTAACATCCTGAAATGGAACG
GTAACGAAGGTGACATCACCGGTCGTGACAAAAACCAGGTTGGTGGTAAAAACTTCTTCAAA
GACGTTGCGCCGTGCACCGACTCTCCGGAATTTTACGGTAAATTCAAAGAAGGTGTTGCGTC
TGGTAACCTGTCTACCGAATTTGAATACACCTTCGACTACCAGATGACCCCGACCAAAGTTA
CCGTTAAAATGCACCAGGACTCTCACGGTGACTCTTACCAGGTTCACGTTTCTCGTGTTGGA
TCTGGAAGTGGCAGCGGATCAGGCAGCGGATCACTCGAGACACCTACGCCGGGTCCGGATCT
GAATGTGGATAATGACCTGCGACCGGAGGCGGGTAGCTACATTGCGAACCTTGCAGCAGCGA
ATACCATGTTCACCACGCGTCTGCATGAGCGTCTGGGTAATACGTACTATACCGACATGGTG
ACGGGTGAGCAGAAACAAACCACTATGTGGATGCGCCATGAAGGTGGTCATAATAAATGGCG
TGATGGCAGCGGCCAGCTGAAAACCCAAAGCAATCGCTATGTTCTGCAACTGGGAGGCGATG
TCGCGCAGTGGAGCCAAAACGGCAGCGACCGCTGGCATGTTGGGGTCATGGCGGGATATGGC
AACAGCGACAGCAAAACCATTTCCTCGCGAACCGGTTATCGTGCAAAAGCGAGTGTGAACGG
ATATAGCACAGGCCTCTATGCCACCTGGTATGCCGATGACGAGTCGCGTAATGGCGCGTATC
TCGACAGTTGGGCGCAGTACAGCTGGTTTGATAACACAGTGAAAGGGGATGACTTACAAAGT
GAATCCTATAAATCAAAAGGATTTACCGCTTCACTGGAAGCTGGATACAAACACAAATTAGC
TGAATTTAATGGCAGCCAGGGAACGCGTAATGAATGGTATGTTCAGCCGCAAGCACAGGTTA
CCTGGATGGGAGTCAAAGCCGATAAGCACCGCGAAAGCAACGGAACCCTCGTTCATAGCAAC
GGTGATGGCAATGTTCAAACCCGACTTGGCGTAAAAACCTGGCTGAAGAGCCACCATAAAAT
GGATGACGGTAAATCCCGCGAGTTCCAGCCGTTTGTAGAAGTGAACTGGCTACATAACAGTA
AGGATTTCAGCACCAGTATGGATGGCGTGTCTGTCACTCAGGATGGAGCCCGAAATATTGCT
GAGATAAAAACCGGGGTGGAAGGACAGCTAAATGCCAACCTGAATGTCTGGGGGAATGTGGG
CGTTCAGGTTGCCGATAGGGGATATAATGACACCTCTGCAATGGTTGGCATTAAGTGGCAAT
TCTGA
```

A.5 nPYP-AT gene sequence

```
MKYLLPTAAAGLLLLAAQPAMAMRGSHHHHHHGSVDGSGSMEHVAFGSEDIENTLAKMDDGQ
LDGLAFGAIQLDGDGNILQYNAAEGDITGRDPKQVIGKNFFKDVAPCTDSPEFYGKFKEGVA
SGNLNTMFEYTFDYQMTPTKVKVHMKKALSGDSYWVFVKRVSGSGSLETPTPGPDLNVDNDL
RPEAGSYIANLAAANTMFTTRLHERLGNTYYTDMVTGEQKQTTMWMRHEGGHNKWRDGSGQL
KTQSNRYVLQLGGDVAQWSQNGSDRWHVGVMAGYGNSDSKTISSRTGYRAKASVNGYSTGLY
ATWYADDESRNGAYLDSWAQYSWFDNTVKGDDLQSESYKSKGFTASLEAGYKHKLAEFNGSQ
GTRNEWYVQPQAQVTWMGVKADKHRESNGTLVHSNGDGNVQTRLGVKTWLKSHHKMDDGKSR
EFQPFVEVNWLHNSKDFSTSMDGVSVTQDGARNIAEIKTGVEGQLNANLNVWGNVGVQVADR
GYNDTSAMVGIKWQF
```

```
ATGAAATACCTGCTGCCGACCGCTGCTGCTGGTCTGCTGCTCCTCGCTGCCCAGCCGGCGAT
GGCCATGAGAGGATCGCATCACCATCACCATCACGGATCCGTCGACGGTAGCGGTTCCATGG
AACATGTGGCGTTTGGCAGCGAAGATATTGAAAACACCCTGGCGAAAATGGATGATGGCCAG
CTGGATGGCCTGGCGTTTGGCGCGATTCAGCTGGATGGCGATGGCAACATTCTGCAGTATAA
CGCGGCGGAAGGCGATATTACCGGCCGCGATCCGAAACAGGTGATTGGCAAAAACTTTTTTA
AAGATGTGGCGCCGTGCACCGATAGCCCGGAATTTTATGGCAAATTTAAAGAAGGCGTGGCG
AGCGGCAACCTGAACACCATGTTTGAATATACCTTTGATTATCAGATGACCCCGACCAAAGT
GAAAGTGCATATGAAAAAAGCGCTGAGCGGCGATAGCTATTGGGTGTTTGTGAAACGCGTGT
CAGGCAGCGGATCACTCGAGACACCTACGCCGGGTCCGGATCTGAATGTGGATAATGACCTG
CGACCGGAGGCGGGTAGCTACATTGCGAACCTTGCAGCAGCGAATACCATGTTCACCACGCG
TCTGCATGAGCGTCTGGGTAATACGTACTATACCGACATGGTGACGGGTGAGCAGAAACAAA
CCACTATGTGGATGCGCCATGAAGGTGGTCATAATAAATGGCGTGATGGCAGCGGCCAGCTG
AAAACCCAAAGCAATCGCTATGTTCTGCAACTGGGAGGCGATGTCGCGCAGTGGAGCCAAAA
CGGCAGCGACCGCTGGCATGTTGGGGTCATGGCGGGATATGGCAACAGCGACAGCAAAACCA
TTTCCTCGCGAACCGGTTATCGTGCAAAAGCGAGTGTGAACGGATATAGCACAGGCCTCTAT
GCCACCTGGTATGCCGATGACGAGTCGCGTAATGGCGCGTATCTCGACAGTTGGGCGCAGTA
CAGCTGGTTTGATAACACAGTGAAAGGGGATGACTTACAAAGTGAATCCTATAAATCAAAAG
GATTTACCGCTTCACTGGAAGCTGGATACAAACACAAATTAGCTGAATTTAATGGCAGCCAG
GGAACGCGTAATGAATGGTATGTTCAGCCGCAAGCACAGGTTACCTGGATGGGAGTCAAAGC
CGATAAGCACCGCGAAAGCAACGGAACCCTCGTTCATAGCAACGGTGATGGCAATGTTCAAA
CCCGACTTGGCGTAAAAACCTGGCTGAAGAGCCACCATAAAATGGATGACGGTAAATCCCGC
GAGTTCCAGCCGTTTGTAGAAGTGAACTGGCTACATAACAGTAAGGATTTCAGCACCAGTAT
GGATGGCGTGTCTGTCACTCAGGATGGAGCCCGAAATATTGCTGAGATAAAAACCGGGGTGG
AAGGACAGCTAAATGCCAACCTGAATGTCTGGGGGAATGTGGGCGTTCAGGTTGCCGATAGG
GGATATAATGACACCTCTGCAATGGTTGGCATTAAGTGGCAATTCTGA
```

A.6 △c-mPYP-AT gene sequence

```
MKYLLPTAAAGLLLLAAQPAMAMRGSHHHHHHGSVDGSGSGVVILDGDGNILKWNGNEGDIT
GRDKNQVGGKNFFKDVAPCTDSPEFYGKFKEGVASGNLSTEFEYTFDYQMTPTKVTVKMHQD
SHGDSYQVHVSRVGSGSGSGSGSGSLETPTPGPDLNVDNDLRPEAGSYIANLAAANTMFTTR
LHERLGNTYYTDMVTGEQKQTTMWMRHEGGHNKWRDGSGQLKTQSNRYVLQLGGDVAQWSQN
GSDRWHVGVMAGYGNSDSKTISSRTGYRAKASVNGYSTGLYATWYADDESRNGAYLDSWAQY
SWFDNTVKGDDLQSESYKSKGFTASLEAGYKHKLAEFNGSQGTRNEWYVQPQAQVTWMGVKA
DKHRESNGTLVHSNGDGNVQTRLGVKTWLKSHHKMDDGKSREFQPFVEVNWLHNSKDFSTSM
DGVSVTQDGARNIAEIKTGVEGQLNANLNVWGNVGVQVADRGYNDTSAMVGIKWQF
```

```
ATGAAATACCTGCTGCCGACCGCTGCTGCTGGTCTGCTGCTCCTCGCTGCCCAGCCGGCGAT
GGCCATGAGAGGATCGCATCACCATCACCATCACGGATCCGTCGACGGTAGCGGTTCCGGTG
TTGTTATCCTGGACGGTGACGGTAACATCCTGAAATGGAACGGTAACGAAGGTGACATCACC
GGTCGTGACAAAAACCAGGTTGGTGGTAAAAACTTCTTCAAAGACGTTGCGCCGTGCACCGA
CTCTCCGGAATTTTACGGTAAATTCAAAGAAGGTGTTGCGTCTGGTAACCTGTCTACCGAAT
TTGAATACACCTTCGACTACCAGATGACCCCGACCAAAGTTACCGTTAAAATGCACCAGGAC
TCTCACGGTGACTCTTACCAGGTTCACGTTTCTCGTGTTGGATCTGGAAGTGGCAGCGGATC
AGGCAGCGGATCACTCGAGACACCTACGCCGGGTCCGGATCTGAATGTGGATAATGACCTGC
GACCGGAGGCGGGTAGCTACATTGCGAACCTTGCAGCAGCGAATACCATGTTCACCACGCGT
CTGCATGAGCGTCTGGGTAATACGTACTATACCGACATGGTGACGGGTGAGCAGAAACAAAC
CACTATGTGGATGCGCCATGAAGGTGGTCATAATAAATGGCGTGATGGCAGCGGCCAGCTGA
AAACCCAAAGCAATCGCTATGTTCTGCAACTGGGAGGCGATGTCGCGCAGTGGAGCCAAAAC
GGCAGCGACCGCTGGCATGTTGGGGTCATGGCGGGATATGGCAACAGCGACAGCAAAACCAT
TTCCTCGCGAACCGGTTATCGTGCAAAAGCGAGTGTGAACGGATATAGCACAGGCCTCTATG
CCACCTGGTATGCCGATGACGAGTCGCGTAATGGCGCGTATCTCGACAGTTGGGCGCAGTAC
AGCTGGTTTGATAACACAGTGAAAGGGGATGACTTACAAAGTGAATCCTATAAATCAAAAGG
ATTTACCGCTTCACTGGAAGCTGGATACAAACACAAATTAGCTGAATTTAATGGCAGCCAGG
GAACGCGTAATGAATGGTATGTTCAGCCGCAAGCACAGGTTACCTGGATGGGAGTCAAAGCC
GATAAGCACCGCGAAAGCAACGGAACCCTCGTTCATAGCAACGGTGATGGCAATGTTCAAAC
CCGACTTGGCGTAAAAACCTGGCTGAAGAGCCACCATAAAATGGATGACGGTAAATCCCGCG
AGTTCCAGCCGTTTGTAGAAGTGAACTGGCTACATAACAGTAAGGATTTCAGCACCAGTATG
GATGGCGTGTCTGTCACTCAGGATGGAGCCCGAAATATTGCTGAGATAAAAACCGGGGTGGA
AGGACAGCTAAATGCCAACCTGAATGTCTGGGGGAATGTGGGCGTTCAGGTTGCCGATAGGG
GATATAATGACACCTCTGCAATGGTTGGCATTAAGTGGCAATTCTGA
```

A.7 △c-nPYP-AT gene sequence

```
MKYLLPTAAAGLLLLAAQPAMAMRGSHHHHHHGSVDGSGSGAIQLDGDGNILQYNAAEGDIT
GRDPKQVIGKNFFKDVAPCTDSPEFYGKFKEGVASGNLNTMFEYTFDYQMTPTKVKVHMKKA
LSGDSYWVFVKRVSGSGSLETPTPGPDLNVDNDLRPEAGSYIANLAAANTMFTTRLHERLGN
TYYTDMVTGEQKQTTMWMRHEGGHNKWRDGSGQLKTQSNRYVLQLGGDVAQWSQNGSDRWHV
GVMAGYGNSDSKTISSRTGYRAKASVNGYSTGLYATWYADDESRNGAYLDSWAQYSWFDNTV
KGDDLQSESYKSKGFTASLEAGYKHKLAEFNGSQGTRNEWYVQPQAQVTWMGVKADKHRESN
GTLVHSNGDGNVQTRLGVKTWLKSHHKMDDGKSREFQPFVEVNWLHNSKDFSTSMDGVSVTQ
DGARNIAEIKTGVEGQLNANLNVWGNVGVQVADRGYNDTSAMVGIKWQF
```

ATGAAATACCTGCTGCCGACCGCTGCTGCTGGTCTGCTGCTCCTCGCTGCCCAGCCGGCGAT
GGCCATGAGAGGATCGCATCACCATCACCATCACGGATCCGTCGACGGTAGCGGTTCCGGCG
CGATTCAGCTGGATGGCGATGGCAACATTCTGCAGTATAACGCGGCGGAAGGCGATATTACC
GGCCGCGATCCGAAACAGGTGATTGGCAAAAACTTTTTTAAAGATGTGGCGCCGTGCACCGA
TAGCCCGGAATTTTATGGCAAATTTAAAGAAGGCGTGGCGAGCGGCAACCTGAACACCATGT
TTGAATATACCTTTGATTATCAGATGACCCCGACCAAAGTGAAAGTGCATATGAAAAAAGCG
CTGAGCGGCGATAGCTATTGGGTGTTTGTGAAACGCGTGTCAGGCAGCGGATCACTCGAGAC
ACCTACGCCGGGTCCGGATCTGAATGTGGATAATGACCTGCGACCGGAGGCGGGTAGCTACA
TTGCGAACCTTGCAGCAGCGAATACCATGTTCACCACGCGTCTGCATGAGCGTCTGGGTAAT
ACGTACTATACCGACATGGTGACGGGTGAGCAGAAACAAACCACTATGTGGATGCGCCATGA
AGGTGGTCATAATAAATGGCGTGATGGCAGCGGCCAGCTGAAAACCCAAAGCAATCGCTATG
TTCTGCAACTGGGAGGCGATGTCGCGCAGTGGAGCCAAAACGGCAGCGACCGCTGGCATGTT
GGGGTCATGGCGGGATATGGCAACAGCGACAGCAAAACCATTTCCTCGCGAACCGGTTATCG
TGCAAAAGCGAGTGTGAACGGATATAGCACAGGCCTCTATGCCACCTGGTATGCCGATGACG
AGTCGCGTAATGGCGCGTATCTCGACAGTTGGGCGCAGTACAGCTGGTTTGATAACACAGTG
AAAGGGGATGACTTACAAAGTGAATCCTATAAATCAAAAGGATTTACCGCTTCACTGGAAGC
TGGATACAAACACAAATTAGCTGAATTTAATGGCAGCCAGGGAACGCGTAATGAATGGTATG
TTCAGCCGCAAGCACAGGTTACCTGGATGGGAGTCAAAGCCGATAAGCACCGCGAAAGCAAC
GGAACCCTCGTTCATAGCAACGGTGATGGCAATGTTCAAACCCGACTTGGCGTAAAAACCTG
GCTGAAGAGCCACCATAAAATGGATGACGGTAAATCCCGCGAGTTCCAGCCGTTTGTAGAAG
TGAACTGGCTACATAACAGTAAGGATTTCAGCACCAGTATGGATGGCGTGTCTGTCACTCAG
GATGGAGCCCGAAATATTGCTGAGATAAAAACCGGGGTGGAAGGACAGCTAAATGCCAACCT
GAATGTCTGGGGGAATGTGGGCGTTCAGGTTGCCGATAGGGGATATAATGACACCTCTGCAA
TGGTTGGCATTAAGTGGCAATTCTGA

Appendix B

ROSETTA DESIGN SCRIPTS

B.1 Mutant structure optimization script

```
<ROSETTASCRIPTS>
        <SCOREFXNS>
            <ScoreFunction name="r15" weights="ref2015" />
        </SCOREFXNS>
        <RESIDUE_SELECTORS>
            <Layer name="surfacelayer" select_core="false"
            ↪  select_boundary="false" select_surface="true"
            ↪  core_cutoff="4.0"/>
            <Index name="n_terminus" resnums="1-27"/>
            <Index name="n_terminus_binding_site"
            ↪  resnums="28-32,41-45,54-55,58,89,91,106-108,⌋
            ↪  110-114,119,121-123"/>
            <Not name="non_binding_or_terminus_residues">
                <Or selectors="n_terminus,⌋
                ↪  n_terminus_binding_site"
                ↪  />
            </Not>
            <Not name="non_terminus_residues"
            ↪  selector="n_terminus"/>
            <Not name="non_binding_residues"
            ↪  selector="n_terminus_binding_site"/>
        </RESIDUE_SELECTORS>
        <TASKOPERATIONS>
            <RestrictToRepacking name="repack_only"/>
            <ReadResfile name="s17_mutant_binding"
            ↪  filename="../input/mutant_design.res"/>
            <ExtraRotamersGeneric name="extrachi" ex1="1"
            ↪  ex2="1" ex1_sample_level="1"
            ↪  ex2_sample_level="1"/>
            <OperateOnResidueSubset
            ↪  name="prevent_surface_repacking"
            ↪  selector="surfacelayer">
                <PreventRepackingRLT />
                # Can also use RestrictToRepackingRLT
```

```
        </OperateOnResidueSubset>
        <OperateOnResidueSubset
        ↪  name="fix_non_binding_or_terminus_residues"
        ↪  selector="non_binding_or_terminus_residues">
            <RestrictToRepackingRLT />
            <PreventRepackingRLT />
        </OperateOnResidueSubset>
        <OperateOnResidueSubset
        ↪  name="fix_non_terminus_residues"
        ↪  selector="non_terminus_residues">
            <RestrictToRepackingRLT />
            <PreventRepackingRLT />
        </OperateOnResidueSubset>
        <OperateOnResidueSubset
        ↪  name="n_terminus_repack_only"
        ↪  selector="n_terminus">
            <RestrictToRepackingRLT/>
        </OperateOnResidueSubset>

    </TASKOPERATIONS>
    <FILTERS>
    </FILTERS>
    <MOVERS>
        #FastRelax name="fast_relax"
        ↪  relaxscript="default">
        #/FastRelax>
        # Design just the N-terminus
        <FastDesign name="design_terminus" scorefxn="r15"
        ↪  task_operations="s17_mutant_binding,extrachi,
        ↪  fix_non_terminus_residues">
            <MoveMap name="redesign_n_terminus">
                <Span begin="1" end="999" chi="false"
                ↪  bb="false"/> # Set all regions far
                ↪  away to no redesign
                <Span begin="1" end="26" chi="true"
                ↪  bb="true"/> # Fully redesign the
                ↪  N-terminus region
                # Keep BB fixed, redesign rotomers in the
                ↪  binding pocket
                # Give residues 26-27 extra flexibility,
                ↪  to accomondate SZ17 mutant
                <Span begin="26" end="27" chi="true"
                ↪  bb="true"/>
```

```
            <Span begin="28" end="32" chi="true"
            ↪  bb="false"/>
            <Span begin="41" end="45" chi="true"
            ↪  bb="false"/>
            <Span begin="54" end="55" chi="true"
            ↪  bb="false"/>
            <Span begin="58" end="58" chi="true"
            ↪  bb="false"/>
            <Span begin="89" end="89" chi="true"
            ↪  bb="false"/>
            <Span begin="91" end="91" chi="true"
            ↪  bb="false"/>
            <Span begin="106" end="108" chi="true"
            ↪  bb="false"/>
            <Span begin="110" end="114" chi="true"
            ↪  bb="false"/>
            <Span begin="119" end="119" chi="true"
            ↪  bb="false"/>
            <Span begin="121" end="123" chi="true"
            ↪  bb="false"/>
        </MoveMap>
    </FastDesign>
    # Follow pack rotamers with a minimization, just
    ↪  of dihederal angles
    <PackRotamersMover name="repack_binding_site"
    ↪  scorefxn="r15" task_operations="extrachi,⌋
    ↪  fix_non_binding_or_terminus_residues,⌋
    ↪  n_terminus_repack_only"/>
    <MinMover name="min_structure" scorefxn="r15"
    ↪  chi="true" bb="true" tolerance=".001">
        <MoveMap name="redesign_n_terminus">
            <Span begin="1" end="999" chi="false"
            ↪  bb="false"/> # Set all regions far
            ↪  away to no redesign
            <Span begin="1" end="26" chi="true"
            ↪  bb="true"/> # Fully redesign the
            ↪  N-terminus region
            # Keep BB fixed, redesign rotomers in the
            ↪  binding pocket
            # Give residues 26-27 extra flexibility,
            ↪  to accomondate SZ17 mutant
            <Span begin="26" end="27" chi="true"
            ↪  bb="true"/>
```

```
            <Span begin="28" end="32" chi="true"
            ↪  bb="false"/>
            <Span begin="41" end="45" chi="true"
            ↪  bb="false"/>
            <Span begin="54" end="55" chi="true"
            ↪  bb="false"/>
            <Span begin="58" end="58" chi="true"
            ↪  bb="false"/>
            <Span begin="89" end="89" chi="true"
            ↪  bb="false"/>
            <Span begin="91" end="91" chi="true"
            ↪  bb="false"/>
            <Span begin="106" end="108" chi="true"
            ↪  bb="false"/>
            <Span begin="110" end="114" chi="true"
            ↪  bb="false"/>
            <Span begin="119" end="119" chi="true"
            ↪  bb="false"/>
            <Span begin="121" end="123" chi="true"
            ↪  bb="false"/>
        </MoveMap>
    </MinMover>
    <FastRelax name="fast_relax" />
    <FastDesign name="design_rotomers" scorefxn="r15"
    ↪  task_operations="s17_mutant_binding,extrachi,
    ↪  fix_non_binding_or_terminus_residues"
    ↪  >
        <MoveMap name="redesign_n_terminus">
            <Span begin="1" end="999" chi="false"
            ↪  bb="false"/> # Set all regions far
            ↪  away to no redesign
            <Span begin="1" end="26" chi="true"
            ↪  bb="true"/> # Fully redesign the
            ↪  N-terminus region
            # Keep BB fixed, redesign rotomers in the
            ↪  binding pocket
            # Give residues 26-27 extra flexibility,
            ↪  to accomondate SZ17 mutant
            <Span begin="26" end="27" chi="true"
            ↪  bb="true"/>
            <Span begin="28" end="32" chi="true"
            ↪  bb="false"/>
```

```
                    <Span begin="41" end="45" chi="true"
                    ↪  bb="false"/>
                    <Span begin="54" end="55" chi="true"
                    ↪  bb="false"/>
                    <Span begin="58" end="58" chi="true"
                    ↪  bb="false"/>
                    <Span begin="89" end="89" chi="true"
                    ↪  bb="false"/>
                    <Span begin="91" end="91" chi="true"
                    ↪  bb="false"/>
                    <Span begin="106" end="108" chi="true"
                    ↪  bb="false"/>
                    <Span begin="110" end="114" chi="true"
                    ↪  bb="false"/>
                    <Span begin="119" end="119" chi="true"
                    ↪  bb="false"/>
                    <Span begin="121" end="123" chi="true"
                    ↪  bb="false"/>
                </MoveMap>
            </FastDesign>
        </MOVERS>
        <APPLY_TO_POSE>
        </APPLY_TO_POSE>
        <PROTOCOLS>
            <Add mover="design_terminus"/>
            <Add mover="repack_binding_site"/>
            <Add mover="min_structure"/>
            <Add mover="fast_relax"/>
        </PROTOCOLS>
        <OUTPUT scorefxn="r15"/>
</ROSETTASCRIPTS>
```

B.2 SZ-∆c-mPYP design constraints

```
# Default: use the given AA and rotomer
NATAA
start
# Sequence alignment, and what residues we want to redesign
# MEHVAFGSEDIENTLAKMDDGQLDGLAF
# LR*RI**LK*KR**LK*KI**LR*EI**
1 A PIKAA L
2 A PIKAA R
```

```
3 A ALLAA
4 A PIKAA R
5 A PIKAA I
6 A ALLAA
7 A ALLAA
8 A PIKAA L
9 A PIKAA K
10 A ALLAA
11 A PIKAA K
12 A PIKAA R
13 A ALLAA
14 A ALLAA
15 A PIKAA L
16 A PIKAA K
17 A ALLAA
18 A PIKAA K
19 A PIKAA I
20 A ALLAA
21 A ALLAA
22 A PIKAA L
23 A PIKAA R
24 A ALLAA
25 A PIKAA E
26 A PIKAA I
27 A ALLAA
28 A ALLAA
# Binding site redesign
29 - 32 A ALLAA
41 - 45 A ALLAA
54 - 55 A ALLAA
58 A ALLAA
89 A ALLAA
91 A ALLAA
106 - 108 A ALLAA
110 - 114 A ALLAA
119 A ALLAA
121 - 123 A ALLAA
```

B.3 Light state stability design script

```
<ROSETTASCRIPTS>
        <SCOREFXNS>
```

```
    <ScoreFunction name="r15" weights="ref2015" />
</SCOREFXNS>
<RESIDUE_SELECTORS>
    # Before prefix added!
    <Index name="n_terminus" resnums="1-28"/>
    <Index name="n_terminus_binding_site"
    ↪  resnums="29-32,41-45,54-55,58,89,91,106-108,
    ↪  110-114,119,121-123"/>
    # After prefix of length 12 added!
    <Index name="extended_n_terminus" resnums="1-40"/>
    <Index name="extended_n_terminus_binding_site"
    ↪  resnums="42,44,53,55-57,66-67,70,101,103,118,
    ↪  120,122-126,131,133,135"/>
    <Index name="only_linker_residues"
    ↪  resnums="38-40"/>
    <Not name="non_binding_residues"
    ↪  selector="extended_n_terminus_binding_site"/>
    <Not name="non_terminus_binding_residues">
        <Or
        ↪  selectors="extended_n_terminus_binding_site,
        ↪  extended_n_terminus"/>
    </Not>
    <Not name="non_terminus_residues"
    ↪  selector="extended_n_terminus"/>
</RESIDUE_SELECTORS>
<TASKOPERATIONS>
    <RestrictToRepacking name="repack_only"/>
    <ExtraRotamersGeneric name="extrachi" ex1="1"
    ↪  ex2="1" ex1_sample_level="1"
    ↪  ex2_sample_level="1"/>
    <OperateOnResidueSubset
    ↪  name="fix_non_binding_residues"
    ↪  selector="non_binding_residues">
        <PreventRepackingRLT />
    </OperateOnResidueSubset>
    <OperateOnResidueSubset
    ↪  name="fix_non_terminus_residues"
    ↪  selector="non_terminus_residues">
        <PreventRepackingRLT />
    </OperateOnResidueSubset>
    <OperateOnResidueSubset
    ↪  name="fix_non_binding_terminus_residues"
    ↪  selector="non_terminus_binding_residues">
```

```
            <PreventRepackingRLT />
        </OperateOnResidueSubset>
        <OperateOnResidueSubset
        ↪  name="repack_terminus_only"
        ↪  selector="extended_n_terminus">
            <RestrictToRepackingRLT />
        </OperateOnResidueSubset>

    </TASKOPERATIONS>
    <FILTERS>
    </FILTERS>
    <MOVERS>
        <AtomTree name="set_up_foldtree"
        ↪  fold_tree_file="../input/docking_foldtree.txt"
        ↪  />
        <FastRelax name="fast_relax" />
        <RemodelMover name="model_sz"
        ↪  blueprint="../input/light_bp"
        ↪  bypass_fragments="False"
        ↪  quick_and_dirty="True"/>
        <Small name="linker_perturb"
        ↪  residue_selector="only_linker_residues"
        ↪  temperature="1" nmoves="300" angle_max="20.0"
        ↪  preserve_detailed_balance="0"/>
        <MinMover name="min_structure" scorefxn="r15"
        ↪  chi="true" bb="true" tolerance=".001">
            <MoveMap name="move_n_terminus">
                <Span begin="1" end="999" chi="false"
                ↪  bb="false"/> # Set all regions far
                ↪  away to no redesign
                <Span begin="1" end="40" chi="true"
                ↪  bb="true"/> # Fully redesign the
                ↪  N-terminus region
            </MoveMap>
        </MinMover>
        <PackRotamersMover name="repack_binding_site"
        ↪  scorefxn="r15" task_operations="extrachi,
        ↪  fix_non_binding_terminus_residues,
        ↪  repack_terminus_only"/>
    </MOVERS>
    <APPLY_TO_POSE>
    </APPLY_TO_POSE>
    <PROTOCOLS>
```

```
            <Add mover="model_sz"/>
            <Add mover="set_up_foldtree"/>
            <Add mover="linker_perturb"/>
            <Add mover="min_structure"/>
            <Add mover="repack_binding_site"/>
            <Add mover="fast_relax"/>
        </PROTOCOLS>
        <OUTPUT scorefxn="r15"/>
</ROSETTASCRIPTS>
```

B.4 Light state decoy verification script

```
<ROSETTASCRIPTS>
        <SCOREFXNS>
            <ScoreFunction name="r15" weights="ref2015" />
        </SCOREFXNS>
        <RESIDUE_SELECTORS>
            # Before prefix added!
            <Index name="n_terminus" resnums="1-28"/>
            <Index name="n_terminus_binding_site"
            ↪  resnums="29-32,41-45,54-55,58,89,91,106-108,⌋
            ↪  110-114,119,121-123"/>
            # After prefix of length 12 added!
            <Index name="extended_n_terminus" resnums="1-40"/>
            <Index name="extended_n_terminus_binding_site"
            ↪  resnums="42,44,53,55-57,66-67,70,101,103,118,⌋
            ↪  120,122-126,131,133,135"/>
            <Index name="only_linker_residues"
            ↪  resnums="38-40"/>
            <Not name="non_binding_residues"
            ↪  selector="extended_n_terminus_binding_site"/>
            <Not name="non_terminus_binding_residues">
                <Or
                ↪  selectors="extended_n_terminus_binding_site,⌋
                ↪  extended_n_terminus"/>
            </Not>
            <Not name="non_terminus_residues"
            ↪  selector="extended_n_terminus"/>
        </RESIDUE_SELECTORS>
        <TASKOPERATIONS>
            <RestrictToRepacking name="repack_only"/>
```

```
        <ExtraRotamersGeneric name="extrachi" ex1="1"
        ↪  ex2="1" ex1_sample_level="1"
        ↪  ex2_sample_level="1"/>
        <OperateOnResidueSubset
        ↪  name="fix_non_binding_residues"
        ↪  selector="non_binding_residues">
            <RestrictToRepackingRLT />
            <PreventRepackingRLT />
        </OperateOnResidueSubset>
        <OperateOnResidueSubset
        ↪  name="fix_non_terminus_residues"
        ↪  selector="non_terminus_residues">
            <RestrictToRepackingRLT />
            <PreventRepackingRLT />
        </OperateOnResidueSubset>
        <OperateOnResidueSubset
        ↪  name="fix_non_binding_terminus_residues"
        ↪  selector="non_terminus_binding_residues">
            <RestrictToRepackingRLT />
            <PreventRepackingRLT />
        </OperateOnResidueSubset>
        <OperateOnResidueSubset
        ↪  name="repack_terminus_only"
        ↪  selector="extended_n_terminus">
            <RestrictToRepackingRLT />
        </OperateOnResidueSubset>

    </TASKOPERATIONS>
    <FILTERS>
    </FILTERS>
    <MOVERS>
        <AtomTree name="set_up_foldtree"
        ↪  fold_tree_file="../input/docking_foldtree.txt"
        ↪  />
        <FastRelax name="fast_relax" />
        <RemodelMover name="model_sz"
        ↪  blueprint="../input/light_bp"
        ↪  bypass_fragments="False"
        ↪  quick_and_dirty="True"/>
        <Small name="linker_perturb"
        ↪  residue_selector="only_linker_residues"
        ↪  temperature="1" nmoves="300" angle_max="20.0"
        ↪  preserve_detailed_balance="0"/>
```

```
            <MinMover name="min_structure" scorefxn="r15"
            ↪  chi="true" bb="true" tolerance=".001">
                <MoveMap name="move_n_terminus">
                    <Span begin="1" end="999" chi="false"
                    ↪  bb="false"/> # Set all regions far
                    ↪  away to no redesign
                    <Span begin="1" end="40" chi="true"
                    ↪  bb="true"/> # Fully redesign the
                    ↪  N-terminus region
                </MoveMap>
            </MinMover>
        </MOVERS>
        <APPLY_TO_POSE>
        </APPLY_TO_POSE>
        <PROTOCOLS>
            <Add mover="model_sz"/>
            <Add mover="set_up_foldtree"/>
            <Add mover="linker_perturb"/>
            <Add mover="min_structure"/>
            <Add mover="fast_relax"/>
        </PROTOCOLS>
        <OUTPUT scorefxn="r15"/>
</ROSETTASCRIPTS>
```

B.5 Dark state decoy verification script

```
<ROSETTASCRIPTS>
        <SCOREFXNS>
            <ScoreFunction name="r15" weights="ref2015" />
        </SCOREFXNS>
        <RESIDUE_SELECTORS>
            # Before prefix added!
            <Index name="n_terminus" resnums="1-28"/>
            <Index name="n_terminus_binding_site"
            ↪  resnums="29-32,41-45,54-55,58,89,91,106-108,
            ↪  110-114,119,121-123"/>
            # After prefix of length 12 added!
            <Index name="extended_n_terminus" resnums="1-40"/>
            <Index name="extended_n_terminus_binding_site"
            ↪  resnums="41-44,53-57,66-67,70,101,103,
            ↪  118-120,122-126,131,133-135"/>
```

```
        <Index name="only_linker_residues"
        ↪  resnums="38-40"/>
        <Not name="non_binding_residues"
        ↪  selector="extended_n_terminus_binding_site"/>
        <Not name="non_terminus_binding_residues">
            <Or
            ↪  selectors="extended_n_terminus_binding_site,
            ↪  extended_n_terminus"/>
        </Not>
        <Not name="non_terminus_residues"
        ↪  selector="extended_n_terminus"/>
    </RESIDUE_SELECTORS>
    <TASKOPERATIONS>
        <RestrictToRepacking name="repack_only"/>
        <ExtraRotamersGeneric name="extrachi" ex1="1"
        ↪  ex2="1" ex1_sample_level="1"
        ↪  ex2_sample_level="1"/>
        <OperateOnResidueSubset
        ↪  name="fix_non_binding_residues"
        ↪  selector="non_binding_residues">
            <RestrictToRepackingRLT />
            <PreventRepackingRLT />
        </OperateOnResidueSubset>
        <OperateOnResidueSubset
        ↪  name="fix_non_terminus_residues"
        ↪  selector="non_terminus_residues">
            <RestrictToRepackingRLT />
            <PreventRepackingRLT />
        </OperateOnResidueSubset>
        <OperateOnResidueSubset
        ↪  name="fix_non_binding_terminus_residues"
        ↪  selector="non_terminus_binding_residues">
            <RestrictToRepackingRLT />
            <PreventRepackingRLT />
        </OperateOnResidueSubset>
        <OperateOnResidueSubset
        ↪  name="repack_terminus_only"
        ↪  selector="extended_n_terminus">
            <RestrictToRepackingRLT />
        </OperateOnResidueSubset>

    </TASKOPERATIONS>
    <FILTERS>
```

```
        </FILTERS>
        <MOVERS>
            <AtomTree name="set_up_foldtree"
            ↪  fold_tree_file="../input/docking_foldtree.txt"
            ↪  />
            <FastRelax name="fast_relax" />
            <RemodelMover name="model_sz"
            ↪  blueprint="../input/dark_bp"
            ↪  bypass_fragments="False"
            ↪  quick_and_dirty="True"/>
            <Small name="linker_perturb"
            ↪  residue_selector="only_linker_residues"
            ↪  temperature="1" nmoves="300" angle_max="20.0"
            ↪  preserve_detailed_balance="0"/>
            <MinMover name="min_structure" scorefxn="r15"
            ↪  chi="true" bb="true" tolerance=".001">
                <MoveMap name="move_n_terminus">
                    <Span begin="1" end="999" chi="false"
                    ↪  bb="false"/> # Set all regions far
                    ↪  away to no redesign
                    <Span begin="1" end="40" chi="true"
                    ↪  bb="true"/> # Fully redesign the
                    ↪  N-terminus region
                </MoveMap>
            </MinMover>
        </MOVERS>
        <APPLY_TO_POSE>
        </APPLY_TO_POSE>
        <PROTOCOLS>
            <Add mover="model_sz"/>
            <Add mover="set_up_foldtree"/>
            <Add mover="linker_perturb"/>
            <Add mover="min_structure"/>
            <Add mover="fast_relax"/>
        </PROTOCOLS>
        <OUTPUT scorefxn="r15"/>
</ROSETTASCRIPTS>
```

B.6 Docked threading foldtree

```
FOLD_TREE EDGE 40 1 -1 EDGE 40 137 -1
```

B.7 Light state threading blueprint

```
0 x I PIKAA N
0 x I PIKAA E
0 x I PIKAA K
0 x I PIKAA E
0 x I PIKAA E
0 x I PIKAA L
0 x I PIKAA K
0 x I PIKAA S
0 x I PIKAA K
0 x I PIKAA K
0 x I PIKAA A
0 x I PIKAA E
0 x I PIKAA L
0 x I PIKAA R
0 x I PIKAA D
0 x I PIKAA R
0 x I PIKAA I
0 x I PIKAA F
0 x I PIKAA G
0 x I PIKAA L
0 x I PIKAA K
0 x I PIKAA D
0 x I PIKAA K
0 x I PIKAA R
0 x I PIKAA E
0 x I PIKAA L
0 x I PIKAA L
0 x I PIKAA K
0 x I PIKAA D
0 x I PIKAA K
0 x I PIKAA I
0 x I PIKAA D
0 x I PIKAA E
0 x I PIKAA L
0 x I PIKAA R
0 x I PIKAA N
0 x I PIKAA E
1 L I PIKAA I
2 A L PIKAA S
3 F L PIKAA F
```

```
4 G L PIKAA G
5 A . PIKAA V
6 I . PIKAA V
7 Q . PIKAA E
8 L .
9 D .
10 G .
11 D .
12 G .
13 N .
14 I .
15 L .
16 Q . PIKAA Y
17 Y . PIKAA W
18 N .
19 A .
20 A .
21 E .
22 G .
23 D .
24 I .
25 T .
26 G .
27 R .
28 D .
29 P .
30 K .
31 Q .
32 V .
33 I .
34 G .
35 K .
36 N .
37 F .
38 F .
39 K .
40 D .
41 V .
42 A .
43 P .
44 C .
45 T .
46 D .
```

```
47 S .
48 P .
49 E .
50 F .
51 Y .
52 G .
53 K .
54 F .
55 K .
56 E .
57 G .
58 V .
59 A .
60 S .
61 G .
62 N .
63 L .
64 N . PIKAA D
65 T .
66 M .
67 F .
68 E .
69 Y .
70 T .
71 F .
72 D .
73 Y .
74 Q .
75 M .
76 T .
77 P .
78 T .
79 K .
80 V .
81 K . PIKAA L
82 V .
83 H .
84 M .
85 K . PIKAA L
86 K .
87 A .
88 L .
89 S . PIKAA T
```

```
90 G .
91 D .
92 S .
93 Y .
94 W . PIKAA F
95 V .
96 F . PIKAA D
97 V .
98 K . PIKAA T
99 R .
100 V .
```

B.8 Dark state threading blueprint

```
0 x I PIKAA N
0 x I PIKAA E
0 x I PIKAA K
0 x I PIKAA E
0 x I PIKAA E
0 x I PIKAA L
0 x I PIKAA K
0 x I PIKAA S
0 x I PIKAA K
0 x I PIKAA K
0 x I PIKAA A
0 x I PIKAA E
1 M I PIKAA L
2 E I PIKAA R
3 H I PIKAA D
4 V I PIKAA R
5 A I PIKAA I
6 F I PIKAA F
7 G I PIKAA G
8 S I PIKAA L
9 E I PIKAA K
10 D I PIKAA D
11 I I PIKAA K
12 E I PIKAA R
13 N I PIKAA E
14 T I PIKAA L
15 L I PIKAA L
16 A I PIKAA K
```

```
17 K I PIKAA D
18 M I PIKAA K
19 D I PIKAA I
20 D I PIKAA D
21 G I PIKAA E
22 Q I PIKAA L
23 L I PIKAA R
24 D I PIKAA N
25 G I PIKAA E
26 L I PIKAA I
27 A L PIKAA S
28 F L PIKAA F
29 G L PIKAA G
30 A . PIKAA V
31 I . PIKAA V
32 Q . PIKAA I
33 L .
34 D .
35 G .
36 D .
37 G .
38 N .
39 I .
40 L .
41 Q . PIKAA K
42 Y . PIKAA W
43 N .
44 A . PIKAA G
45 A . PIKAA N
46 E .
47 G .
48 D .
49 I .
50 T .
51 G .
52 R .
53 D .
54 P . PIKAA K
55 K . PIKAA N
56 Q .
57 V .
58 I . PIKAA G
59 G .
```

```
60 K .
61 N .
62 F .
63 F .
64 K .
65 D .
66 V .
67 A .
68 P .
69 C .
70 T .
71 D .
72 S .
73 P .
74 E .
75 F .
76 Y .
77 G .
78 K .
79 F .
80 K .
81 E .
82 G .
83 V .
84 A .
85 S .
86 G .
87 N .
88 L .
89 N . PIKAA S
90 T .
91 M . PIKAA E
92 F .
93 E .
94 Y .
95 T .
96 F .
97 D .
98 Y .
99 Q .
100 M .
101 T .
102 P .
```

```
103 T .
104 K .
105 V .
106 K . PIKAA T
107 V .
108 H . PIKAA K
109 M .
110 K . PIKAA H
111 K . PIKAA Q
112 A . PIKAA D
113 L . PIKAA S
114 S . PIKAA H
115 G .
116 D .
117 S .
118 Y .
119 W . PIKAA Q
120 V .
121 F . PIKAA H
122 V .
123 K . PIKAA S
124 R .
125 V .
```

Appendix C

ANALYSIS CODE

C.1 Clustering script

```
#!/usr/bin/python3
from Bio.PDB import PDBParser, Superimposer
import random
import argparse
import os

parser = argparse.ArgumentParser(description="Clusters PDBs")
parser.add_argument('-n', type=int, required=True, help='The
↪  number of clusters to make')
parser.add_argument('directories', nargs=argparse.REMAINDER,
↪  help='Folders containing Rosetta runs')

superimposer = Superimposer()
def get_rms(pdb1, pdb2):
    superimposer.set_atoms(list(pdb1.get_atoms()),
    ↪  list(pdb2.get_atoms()))
    return superimposer.rms


if __name__ == '__main__':
    args = parser.parse_args()
    pdbs = []
    pdbs_assign = []
    strict_parser = PDBParser(PERMISSIVE=0)
    for folder in args.directories:
        with open(os.path.join(folder, 'score.sc')) as scores:
            # Consume the header lines
            next(scores)
            next(scores)
            for line in scores:
                vals = line.split()
                score = float(vals[1])
                name = vals[21]
                filename = os.path.join(folder,name) + '.pdb'
```

```
            structure =
            ↪  strict_parser.get_structure('imported_structure',
            ↪
            ↪  filename)[0]
            pdbs.append((folder, name, structure , score))
            pdbs_assign.append(-1)
print('Done loading!')

# Load files into a list of tuples (folder, name, PDB,
↪  energy)
# Run Lloyd's algorithm
n = args.n


# Randomly choose N centers
old_centers_idx = []
centers_idx = random.sample(range(len(pdbs)),n)

old_pdbs_assign = []
while set(old_centers_idx) != set(centers_idx) or
↪  set(old_pdbs_assign) != set(pdbs_assign):
    old_pdbs_assign = list(pdbs_assign)
    centers = [pdbs[idx] for idx in centers_idx]
    print(centers_idx)

    # Step 1, reassign centers
    for idx, pdb in enumerate(pdbs):
        smallest_rms = 99999999999
        smallest_c = -1
        for c_idx, center in enumerate(centers):
            rms = get_rms(pdb[2], center[2])
            if rms < smallest_rms:
                smallest_c = c_idx
                smallest_rms = rms
        pdbs_assign[idx] = smallest_c
    # Step 2, repick centers.
    average_rms_deviation = 0
    new_centers_idx = [-1] * n
    for raw_idx, c_idx in enumerate(centers_idx):
        center = pdbs[c_idx]
        in_cluster_idx = [idx for idx, clust in
        ↪  enumerate(pdbs_assign) if clust == raw_idx]
        num_pick = 10
```

```
            if num_pick > len(in_cluster_idx):
                num_pick = len(in_cluster_idx)
            selected_idx = random.sample(in_cluster_idx,
            ↪  num_pick)

            possible_centers = [c_idx]
            possible_centers += selected_idx

            best_meta_c_idx = -1
            best_rms = 9e99
            for pcenter in possible_centers:
                rms = 0
                for pdb_idx in in_cluster_idx:
                    rms += get_rms(pdbs[pcenter][2],
                    ↪  pdbs[pdb_idx][2]) ** 2
                if rms < best_rms:
                    best_meta_c_idx = pcenter
                    best_rms = rms
            average_rms_deviation += best_rms
            new_centers_idx[raw_idx] = best_meta_c_idx
        print('Round average
        ↪  deviation:{}'.format(average_rms_deviation / n))
        # Reset centers
        old_centers_idx = list(centers_idx)
        centers_idx = list(new_centers_idx)
    print(centers_idx)
    print('Done clustering!')
    print('Cluster assignments:{}'.format(pdbs_assign))
    centers_processed = [-1] * n
    for raw_idx, center in enumerate(centers_idx):
        members = [pdbs[idx] for idx, clust in
        ↪  enumerate(pdbs_assign) if clust == raw_idx]
        avg_energy = sum([v[3] for v in members]) /
        ↪  len(members)
        centers_processed[raw_idx] = (pdbs[center], members,
        ↪  avg_energy)
    for center in sorted(centers_processed, key=lambda v:
    ↪  v[2]):
        print('Center name:{}/{}\n\tAverage
        ↪  energy:{:2f}\n\tMembers:'.format(center[0][0],
        ↪  center[0][1], center[2]))
        for member in sorted(center[1], key=lambda v: v[3]):
```

```
            print('\t\tMember
            ↪  ({:2f}):{}/{}'.format(member[3], member[0],
            ↪  member[1]))
```