

CHARACTERIZATION OF INTERNAL DYNAMICS IN VAV1:
METHOD DEVELOPMENT, MUTUAL COUPLING
AND FUNCTIONAL RELEVANCE

APPROVED BY SUPERVISORY COMMITTEE

Michael K. Rosen, Ph.D.

Kevin H. Gardner, Ph.D.

Rama Ranganathan, M.D., Ph.D.

Stephen Sprang, Ph.D.

This thesis is dedicated to my family

CHARACTERIZATION OF INTERNAL DYNAMICS IN VAV1:
METHOD DEVELOPMENT, MUTUAL COUPLING
AND FUNCTIONAL RELEVANCE

by

PILONG LI

DISSERTATION

Presented to the Faculty of the Graduate School of Biomedical Sciences

The University of Texas Southwestern Medical Center at Dallas

In Partial Fulfillment of the Requirements

For the Degree of

DOCTOR OF PHILOSOPHY

The University of Texas Southwestern Medical Center at Dallas

Dallas, Texas

May, 2009

Copyright

by

PILONG LI, 2009

All Rights Reserved

ACKNOWLEDGEMENTS

I am thankful to many people who make my Ph. D career at UTSW possible and fruitful. First and for most, I have the sincerest gratitude to my mentor, Dr. Michael K. Rosen. I walked into his office to request a rotation opportunity early March, 2003. Although actin regulation has been Mike's expertise, I was too ignorant to know what actin was at that time. Surprisingly Mike gave me the rotation opportunity and then accepted me as his graduate student a few months later. During the subsequent years, Mike has always challenged me with cutting-edge projects. He himself has provided me a true model as a scholar: fearlessly uncovering scientific truth, rigorously treating experimental data, leaving no stone unturned, etc. With his patience, wisdom, support and challenge, I am equipped sufficient knowledge and scientific skills to think creatively and to tackle interesting questions on a solid ground. In my following scientific career, Mike will always be my model. Whenever there are obstacles facing me, I would ask myself "What will Mike do if he were me".

I have had the luxury of interacting with an extremely supportive thesis committee including Dr. Kevin Gardner, Dr. Rama Ranganathan, and Dr. Steve Sprang. I thank them for giving their time, energy, and insightful comments throughout the course of my work.

Over the years, all past and current Rosen lab members have helped me in many ways. They are the hidden heroes/heroines behind my success. Dr. Gaya Amarasinghe was my instructor during my rotation in the lab. It was under his guidance that I had my first intimate touch with NMR magnets and then fell in "love" with NMR since then. Ilidio Martins, a graduate, joined me in the Vav dynamics project when it was in its down time. We, together, extensively looked for ways to tackle the problems. Whenever I needed help, experimentally or not, he has always generously offered me a hand unconditionally. He has been a perfect collaborator and a great friend indeed. I also had the benefit to work closely with four fantastic graduate students in the lab, Hui-chun Cheng, Bingke Yu, Soyeon Kim and Sudeep Banjade. Their endless energy in doing science is rather contagious, which has been constantly motivating me to move forward. There is a Chinese saying: "There is certainly a teacher of you among any three people". I have a lot good teachers and friends. Drs. Takanori Otomo and Shae Padrick are certainly among these teachers and friends. They taught me, without any reservation, a lot of knowledge, from which my research benefited a lot. I enjoyed late night (early morning to be precise) long discussions with them. I benefit a lot from discussing NMR dynamics with Drs. Xiaolan Yao and David Morgan. I also enjoyed arguing about who is more senior in lab with Ayman Ismail, besides the friendship from him. I enjoy the dynamic and lovely environment created by Junko, Xiaolan and Hui-chun. Drs. Jia Da, Stone Chen, Zhucheng Chen, Xiaocheng Chen, Eduardo Torres, Daisy Leung, Abhi Seth, Sanjay Pantral gave me helps in many ways. Linda Doolittle, Elisha White and Chinatzu Otomo are

the indispensable part of the lab and they make my research as smooth as it can be. Last, I am specially thankful to Dee Seres and Jo Appleton who has been taking care of me in many aspects.

Drs. Carlos Amezcua (UTSW) and Geoff Armstrong's (UCHSC) excellent support in NMR data collection is an integral part of my thesis. I am in debt to them. I am also lucky to have collaborated with three smart and friendly individuals from Rama lab: Dr. Bill Russ, Alan Poole and Dr. Chris Larson. I benefited a lot through interacting with them.

Many of my friends have made my life at UTSW much more fruitful. They are Kunhua Song and his family, Tzu-Ming Wang and his family, Wei Tang and her family, Jake Chen, Hua Lin and her family, all soccer fan friends. They all deserve my sincere gratitude.

My work would not be possible if not for the endless love, support from my parents. They received very limited education themselves. But they know how important for me to receive as much education as possible. They full heartedly created an environment comfortable for me to finishing my primary school, middle school, college, and finally graduate school. There are few far in between people of my background who can accomplish this amazing journey. I can never say enough of them. I also thank my younger brother who is always there for my parents when I am quite some distance away.

There is one person who is most special, my wife. She is the biggest fan of my work although she does not necessarily understand it. She sells my work to all her non-biology major friends as if it were the most amazing work done ever. She wholeheartedly supports any decision I've made even though it mean a lot scarifies from her. I owe her everything.

CHARACTERIZATION OF PROTEIN DYNAMICS IN VAV1:
METHOD DEVELOPMENT, MUTUAL COUPLING
AND FUNCTIONAL RELEVANCE

PILONG LI, Ph.D.

The University of Texas Southwestern Medical Center at Dallas, 2009

Supervising Professor: MICHAEL K. ROSEN, Ph.D.

Protein motions are important to activity, but quantitative relationships between internal dynamics and function are not well understood. The Dbl homology (DH) domain of the proto-oncoprotein and guanine nucleotide exchange factor Vav1 is autoinhibited through interactions between its catalytic surface and a helix from an N-terminal acidic region. Phosphorylation of the helix relieves autoinhibition. Here I show by NMR spectroscopy that the autoinhibited DH domain (AD) exists in equilibrium between a ground state, where the active site is blocked by the inhibitory helix, and an excited state, where the helix is dissociated. Across a series of mutants that differentially sample these states, catalytic activity of the autoinhibited protein and

its rate of phosphorylation are linearly dependent on the population of the excited state. Thus, internal dynamics are required for and control both basal activity and the rate of full activation of the autoinhibited DH domain.

Vav1 belong to a class of multi-domain signaling proteins exhibit complex behaviors due to cooperative interactions between domains. In many such proteins there is a core regulatory interaction, involving binding of an inhibitory element to the active site of a functional domain like the inhibitory helix to DH in Vav1. The core interaction is cooperatively enhanced by additional intramolecular domain-domain contacts. The physical basis of this cooperativity, and thus the energetic construction of multi-domain systems, is not well understood. Dynamics analysis of AD reveals that the closed and open populations are about 10:1 for the core interaction in isolation. In the full five-domain regulatory fragment of Vav1, interactions between domains outside of the core further bias this inhibitory equilibrium ~10-fold toward the closed state, further suppressing activity. Thus, Vav1 is controlled by two, weakly biasing, but thermodynamically coupled equilibria--an energetic construction that is probably general among multi-domain proteins.

The dynamic landscape of AD is composed of two μs -ms time scale motions: one is the inhibitory helix binding to and dissociating from the DH domain and another is intrinsic to the DH domain. Interestingly relative populations and exchange rates of the second process are altered upon perturbations to the inhibitory helix, suggesting that the two dynamic processes are energetically and kinetically coupled. A strategy has been established to quantify the thermodynamic and kinetic coupling strengths between the two processes via direction parameterization of four-state equilibria using NMR Carr-Purcell-Meiboom-Gill measurement. The coupling strengths between the two dynamic processes in AD are 1.0~1.5 kcal M⁻¹

comparable to the coupling strength between the modulatory interaction and the helix-DH interactions in the full five-domain regulatory fragment of Vav1. The coupling strength is relatively weak consistent with the coupling strengths reported for many other signaling proteins such as Src tyrosine kinase. These findings suggest that weakly coupling may be a common theme in regulatory molecules.

TABLE OF CONTENTS

Title Fly	i
Dedication	ii
Title Page	iii
Copyright	iv
Acknowledgements.....	v
Abstract	vii
Table of Contents.....	x
Publications	xii
List of Figures.....	xiii
List of Tables	xv
List of Derivations	xvi
List of Abbreviations	xvii
Chapter 1 Introduction	1
Vav biology.....	4
Vav biochemistry	7
Pre-equilibrium view of autoinhibition and its role in activation.....	13
Cooperative autoinhibition.....	15
Experimental approaches to study protein dynamics.....	21
Functional Relevance of Microsecond-millisecond Dynamics	25
Qualitative correlations between dynamics and function	26
Pre-equilibria in binding and catalysis.....	27
Quantitative correlations between dynamics and function	29
Co-existing Multiple Dynamic Processes.....	33
Chapter 2 Internal dynamic control activation and activity of the autoinhibited Vav DH domain	39
Introductions	39
Results	42
The AD module undergoes motions on the μ s-ms timescale	42
The inhibitory arm fluctuates between bound and free states.....	48
Quantitative analysis of dynamics in the AD protein	51
Phosphorylation occurs through the excited state.....	57
Helix-DH conformational equilibrium controls GEF activity	61
CH interactions cooperatively suppress DH GEF activity in CADPZ	63
Discussion	66
Methods	71
Chapter 3 Theoretical Assessment of Parameterization of Four-state Equilibrium Using NMR	77
Introduction.....	77
Results and Discussion	81
The four-state equilibrium model in AD	81
Global solution determination.....	83

χ^2 surface mapping	86
Effects of incorporation of orthogonal information.....	90
Global solution determination in empirical application.....	92
Robustness of the global solution	94
Feasibility in major parameter space	96
Improvement upon addition of more measurements	101
Conclusions.....	104
Materials and Methods.....	105
Chapter 4 Allosteric Coupling Strengths	
between Two Dynamic Processes	111
Introduction.....	111
Results	115
The four-state model in the Vav1 DH domain.....	115
Parameterization of the four-state equilibrium	117
Robustness of the global solution	123
Model Selection	123
CPMG curve fitting recapitulates	
open populations measured independently	128
Energetic and kinetic coupling strengths	130
Discussion	133
Materials and methods	135
Chapter 5 Concluding remarks	141
References	147
Appendix 1 Mathematica code for generating relaxation dispersion	
data for four-state equilibrium	160
Appendix 2 Step by step CPMG relaxation dispersion data analyses	164
Appendix 3 get_spectra.pl.....	174
Appendix 4 fid_com.pl.....	179
Appendix 5 proc_com.pl	181
Appendix 6 process2DSpe.pl.....	192
Appendix 7 R2calculation.pl.....	202
Appendix 8 inputgeneration.pl	208
Appendix 9 generatecpmgdata.pl	212
Appendix 10 curvefitting.pl.....	216
Appendix 11 C code for curve fitting	222
Appendix 12 List of Fortran modules used in Appendix 11	282
Appendix 13 Makefile script I.....	283
Appendix 14 Makefile script II	284

PRIOR PUBLICATIONS

Celestine J. Thomas, Xinlin Du, **Pilong Li**, Ying Wang, Elliott M. Ross, Stephen R. Sprang
Uncoupling conformational change from GTP hydrolysis in a heterotrimeric G protein alpha-subunit. Proceedings of the National Academy of Sciences of the United States of America
2004;101(20):7560-5.

Pilong Li, Ilídio R. S. Martins, Gaya K. Amarasinghe, Michael K. Rosen

Internal dynamics control activation and activity of the autoinhibited Vav DH domain.

Nature structural & molecular biology 2008;15(6):613-8.

LIST OF FIGURES

Figure 1-1	Domain architecture of Vav1 proteins.	3
Figure 1-2	Different truncations of Vav.....	9
Figure 1-3	Structural basis of Vav1 autoinhibition.....	10
Figure 1-4	Structural models for Tim, Ngef and Wgef.....	17
Figure 1-5	Time scales for NMR dynamics techniques.....	24
Figure 1-6	Thermodynamic models for two-, linear three-, circular three- and four-state	34
Figure 2-1	Relaxation dispersion in AD and structural mapping of dynamic residues.....	44
Figure 2-2	Single quantum ^{13}C methyl relaxation dispersion curves for representative resonances that show increased relaxation dispersion amplitude upon phosphorylation.....	45
Figure 2-3	Ribbon diagram of the pAD structured model based on AD solution structure with methyl-containing sidechains showing $\Delta R_2 > 2.0 \text{ s}^{-1}$ drawn as sticks.....	46
Figure 2-4	Single quantum ^{13}C methyl relaxation dispersion curves for pAD and the $\Delta\text{A-D}$ and $\Delta\text{A-D}_{\text{K208E}}$	47
Figure 2-5	Models for the four-state equilibrium and the simplified two-state equilibrium involving the Ac helix in the AD protein.	49
Figure 2-6	AD mutants differentially sample the helix-DH equilibrium.....	53
Figure 2-7	Overlay of $^1\text{H}/^{13}\text{C}$ HSQC spectra.	54
Figure 2-8	Vav mutants show different phosphorylation rates.	59
Figure 2-9	Conformational equilibrium of the inhibitory helix controls the rate of AD phosphorylation.	60
Figure 2-10	Conformational equilibrium of the inhibitory helix controls GEF activity of AD and CADPZ.....	62
Figure 2-11	Interdomain cooperativity in Vav1.....	65
Figure 2-12	Model for function and regulation of the AD module of Vav1.....	67
Figure 2-13	Domain architecture of CADPZ and thermodynamic model for its cooperative suppression.	70
Figure 3-1	The four- and two-state models.	80
Figure 3-2	Hypothetical noise-free and noise-incorporated target function surfaces	84
Figure 3-3	Curve fitting results of synthesized CPMG data.	88
Figure 3-4	Determining the global solution using chemical shift differences.....	93
Figure 3-5	Input kinetic and thermodynamic parameters in parameter space survey.	97
Figure 3-6	Fitted kinetic and thermodynamic parameters in parameter space survey.	99
Figure 3-7	Effects of more measurements on fitted kinetic and thermodynamic parameters.	103
Figure 4-1	The four-state model derived from two processes and	

	the hypothetical energy diagrams of one process.	114
Figure 4-2	Solve the four-state equilibrium of AD.	121
Figure 4-3	Representative CPMG relaxation dispersion curves.	122
Figure 4-4	Monte Carlo simulation results.	124
Figure 4-5	F-test distinguishes the circular 4-state model from other simpler models for AD.	126
Figure 4-6	Correlation between experimentally determined open populations with fitted ones from CPMG.	131

LIST OF TABLES

Table 3-1.	Robustness of the global solution towards random noise.	95
Table 4-1.	Fitted parameters and kinetic and thermodynamic coupling strengthens of AD.	132
Table 5-1.	Multiple weakly biasing interactions cooperatively provide strong suppression.	143

LIST OF DERIVATIONS

Derivations 1. Linear behavior of chemical shift in perturbations of a four-state equilibrium.....	52
Derivations 2. In highly biased equilibria relaxation dispersion is much more sensitive than chemical shift to changes in equilibrium populations.	58

LIST OF ABBRIVATIONS

°C	degrees Celsius
A.U.	arbitrary units
BPTI	basic pancreatic trypsin inhibitor
CPMG	Carr-Purcell-Meiboom-Gill
CypA	Cycliphilin A
DQ	double quantum
DTT	dithiothreitol
E_a	activation energy
GDI	guanine nucleotide inhibitor
GDP	guanosine diphosphate
GEF	guanine nucleotide exchange factor
GTP	guanosine triphosphate
HSQC	heteronuclear spin quantum correlation
IPTG	isopropyl- β -D-thiogalactopyranoside
kcal	kilocalorie
KD	equilibrium dissociation constant
kDa	kiloDalton
LAND	common logarithms of average normalized pair wise distances
mg	milligrams
min	minutes
mol	mole
MQ	multiple quantum

ms	milliseconds
MWC	Monod, Wyman, and Changeux
nm	nanometer
nM	nanomolar
NMR	nuclear magnetic resonance
NOE	Over-houser effect
PIP2	phosphatidylinositol 4,5-bisphosphate
ppm	parts per million
PRE	paramagnetic relaxation enhancement
ps	picoseconds
$R1\rho$	spin relaxation rate constant in rotating frame
RDC	residual dipolar coupling
Rho	Ras homology
sec	seconds
SH2	Src homology 2
SH3	Src homology 3
SQ	single quantum
TROSY	transverse relaxation optimized spectroscopy
ΔG	Gibbs' free energy
μM	micromolar
μs	microseconds
ZQ	zero quantum

Chapter 1 Introduction

Multi-domain signaling proteins are sophisticated “machines” which receive, often complicated, input signals, do proper interpretations and then orchestrate downstream performance accordingly. Each machine requires proper operational modes to fulfill its routine functions. Multi-domain signaling proteins are no exceptions. Clear deciphering of the operational modes within these “machines” is crucial for our understanding about how they can ensure necessary intricacy and fidelity for signal transduction *in vivo*. Operational modes in proteins referred to in this thesis are the physical state(s) of proteins that are functional, the populations of such states, how these states are achieved and how fast they interconvert. Clearly these operational modes are encoded in the structural snapshots and in the kinetics and thermodynamics within and between the snapshots. Much progress has been made towards structural characterization of the snapshots thanks to mature X-ray crystallographic and NMR spectroscopic techniques as well as emerging efforts towards atomic resolution electron microscopy. The dynamics (kinetics and thermodynamics) of proteins, especially of multi-domain proteins, however, are left behind due to technical challenges.

This thesis is devoted to decipher operational modes in Vav protooncoprotein, a class of multi-domain signaling proteins. In the process, novel NMR techniques have been developed, which are well suited for deciphering

operational modes embedded in other multi-domain proteins. In Chapter 2, I first investigate the dynamic landscape of a minimally autoinhibited Dbl homology (DH) domain (referred to as AD, see Figure 1-1 for details) of Vav1. In AD, there are two millisecond-microsecond dynamic processes, one of which is consistent with the scenario that the inhibitory element is constantly binding to and dissociating from the DH domain. Using chemical shift perturbation analysis on a series of construct, I demonstrate that the abovementioned dynamic process controls the basal activity of AD and its activation rate by Src family tyrosine kinase. I then further demonstrate that this dynamic process also exists in the full regulatory apparatus of Vav1, CADPZ (see Figure 1-1 for details about CADPZ). Importantly it is further modulated ~ 10 -fold toward the autoinhibited/closed state by other domain-domain interactions in CADPZ. In Chapter 3, I use computer simulation to establish a strategy to solve four-state equilibria derived from two dynamic processes. In Chapter 4, the strategy is used to solve the four-state equilibrium derived from the two dynamic processes observed in AD.

In this chapter, I first summarize the existing literature about Vav and infer the potential operational modes in Vav. The operational modes are generalized by comparing Vav with other multi-domain signaling proteins. Then I review techniques that are central to deciphering operational modes and emphasis is placed on NMR techniques providing information μ s-ms motions. The current view of and evidence about links between microsecond-millisecond

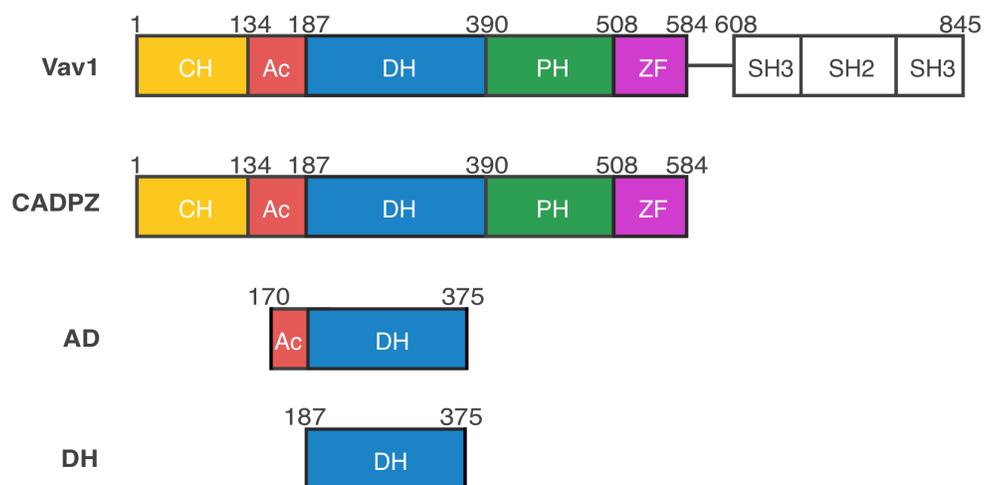


Figure 1-1. Domain architecture of Vav1 proteins. Domains are colored: CH (orange), Ac (red), DH (blue), PH (green), ZF (magenta), SH3-SH2-SH3 (white). The first five domains forms the full regulatory apparatus (CADPZ) and the SH3-SH2-SH3 module is responsible for localization. The number corresponds the number in mouse Vav1. The DH domain bears guanine nucleotide exchange activity. It is auto-inhibited by directing binding of an helix derived from the C-terminus of Ac region. The structural basis of the autoinhibition is demonstrated in a construct containing the

motions and functions are summarized. Last, I summarize emerging evidence suggesting that it is rule rather than exception for more than one μ s-ms process coexisting in one system.

Vav biology

In 1989, Barbacid and coworkers isolated a novel gene exhibiting transforming activity in a gene transfer assay screening for potential human oncogene (Katzav, Martin-Zanca et al. 1989). Since this gene was the sixth oncogene identified in their laboratory, it was designated by the acronym *vav*, which is the sixth letter of the Hebrew alphabet (Katzav, Martin-Zanca et al. 1989). Soon after that, the full length mouse *vav* proto-oncogene was isolated in the same laboratory (Coppola, Bryant et al. 1991) and the previously identified gene lacks the genetic codes for the first 65 amino acids. *Vav2*, a *Vav* homolog, was isolated in human and mouse (Henske, Short et al. 1995; Schuebel, Bustelo et al. 1996) and the originally discovered *Vav* has since been referred to as *Vav1*. A decade after the discovery of *Vav1*, Movilla and Bustelo isolated *Vav3*, the third and last member of *Vav* proteins in mammals (Movilla and Bustelo 1999). While *Vav1* is primarily expressed in cells of hematopoietic lineage (Ogilvy, Elefanty et al. 1998), *Vav2* and *Vav3* are ubiquitously expressed (Hornstein, Alcover et al. 2004).

The proto-oncoproteins *Vav1*, 2 and 3 are important players in mediating signals from a variety of cell surface receptors, for instance T-cell receptor

(Bustelo 2001; Tybulewicz, Ardouin et al. 2003; Tybulewicz 2005) including CD46/CD3 costimulatory interactions (Zaffran, Destaing et al. 2001), B-cell receptor (Bachmann, Nitschke et al. 1999; Bustelo 2001; Tybulewicz 2005), Fc γ (Bustelo 2001; Tybulewicz 2005), integrin receptor (Gotoh, Takahira et al. 1997; Gakidis, Cullere et al. 2004), and cytokine receptors (Kishimoto, Taga et al. 1994) including epidermal growth factor receptor (Pandey, Podtelejnikov et al. 2000; Tamas, Solti et al. 2001; Tamas, Solti et al. 2003), platelet-derived growth factor receptor (Pandey, Podtelejnikov et al. 2000), and G-protein coupled receptors (Kim, Marchal et al. 2003), to the interior of the cells.

Vav proteins contain a domain sharing homology with dbl homology domain of dbl, Bcr, and Cdc24 (Adams, Houston et al. 1992; Cen, Papageorge et al. 1992; Galland, Katzav et al. 1992) (Fig. 1-1) indicating that Vav functions as guanine nucleotide exchange factors (GEFs) for Rho family small GTPase (Billadeau 2002) as postulated by Puil and Pawson *a priori* (Puil and Pawson 1992). The Rho family small GTPases, Cdc42, Rac and Rho are well-established regulators of fundamental cellular processes including cytoskeletal dynamics, transcription and cell proliferation (Schmitz, Govek et al. 2000; Burridge and Wennerberg 2004; Arimura and Kaibuchi 2005; Decker, Moon et al. 2005; Sorokina and Chernoff 2005). Therefore Vav proteins are responsible for relaying signals from cell surface receptors to regulate cytoskeleton rearrangement and transcriptional modulation (Katzav 2004).

Vav1 plays important roles in signal transduction in the immune systems (Katzav 2007). For example it controls functions such as T cell activation, phagocytosis by macrophages, and superoxide production by neutrophils (Hall et al., 2006; Tybulewicz, 2005; Utomo et al., 2006). Vav1 also plays a critical role in hematopoietic cells development from totipotent cells (Wulf, Adra et al. 1993). Vav1 null mice exhibit low number of T cell and/or B cell (Fischer, Zmuldzinas et al. 1995; Tarakhovsky, Turner et al. 1995; Zhang, Alt et al. 1995; Fujikawa, Miletic et al. 2003). Vav2 and Vav3 also function outside of immune systems. For example, Vav2 is involved in axonal guidance (Cowan, Shao et al. 2005). Aberrant Vav signaling is observed in, or contributes to, many cellular abnormalities and diseases. N-terminal truncation of Vav leads to cell transformation (Katzav, Martin-Zanca et al. 1989; Abe, Whitehead et al. 1999; Zeng, Sachdev et al. 2000; Booden, Campbell et al. 2002). Vav1 upregulation in pancreatic adenocarcinoma associates with decreased patient survival (Fernandez-Zapico, Gonzalez-Paz et al. 2005). Vav1 deficiency is correlated with the human primary immune disorder, common variable immunodeficiency, and is suggested to play an important role in the pathogenesis of this disease (Paccani, Boncristiano et al. 2005). Vav1 is also implicated in HIV infection and AIDS. For example, Vav1 deficiency protects mice against replication of murine leukemia virus and slows progression to symptoms of murine AIDS (Knoetig, Torrey et al. 2002). In addition, Vav1 is activated by HIV Nef and may account

for the ability of Nef to stimulate activation of Rac, Cdc42 and the Pak kinase (a Cdc42/Rac effector), to facilitate HIV pathogenesis (Fackler, Luo et al. 1999; Quaranta, Mattioli et al. 2003). Thus, further detailed knowledge of Vav regulation will contribute to our understanding of these important biological processes, and can lead to new strategies for the diagnosis and treatment of cancer, immune disease and HIV/AIDS.

Vav biochemistry

The Vav proteins are composed of an N-terminal calponin homology (CH) domain (residues 1-134) followed by an acidic region (Ac) (135-187), the Dbl homology (DH) (188-390) and pleckstrin homology (PH) (391-508) domains common to all Rho GEFs (Zheng 2001), a zinc finger (ZF) (509-584) domain, and an SH3-SH2-SH3 (608-845) module at the C-terminus (Fig. 1-1) (Bustelo 1996). Vav possesses a variety of intramolecular interactions inferred from biochemistry and genetics studies, some of which are observed directly from structural studies (Aghazadeh, Lowry et al. 2000; Llorca, Arias-Palomo et al. 2005). These interactions are important for Vav regulation *in vivo*. Truncation of the N-terminal 65 amino acids (N-terminal half of CH domain) of Vav1 (Vav1 Δ 65) activates its transforming activity, suggesting this sequence participates in a negative regulation on Vav function (Crespo, Schuebel et al. 1997; Bustelo 2001) (Fig. 1-2). Deeper truncation of N-terminus of Vav1 to 186 gives rise to Vav1 Δ 186 (CH domain and the whole Ac region are deleted), which is more

potent in cell transforming activity than Vav1 Δ 65 (Crespo, Schuebel et al. 1997; Bustelo 2001). Collectively, these data and other suggest that the N-terminus (1-186) of Vav negatively regulates Vav activity by forming intramolecular interactions and probably multiple interactions (physically and/or thermodynamically) with its C-terminal portion (amino acid 187 and beyond) (Fig. 1-2).

The first structural evidence of physical intramolecular interaction between the N-terminal and the C-terminal part of Vav was provided by an NMR solution structure of the C-terminal half of Ac region with the DH domain of Vav (hereafter called AD containing amino acid 170-375, Fig. 1-3) (Aghazadeh, Lowry et al. 2000). Indeed, the acidic fragment of AD protein forms an amphipathic α -helix and binds in the active site of the DH domain (Fig. 1-3). Such a binding mode excludes GTPase access to the active site and therefore inhibits GEF activity consistent with observation that Vav1 Δ 186 is potent in inducing cell transforming activity. We term the interaction of the inhibitory helix with the DH domain, exemplified by the solution structure of AD, as “core autoinhibitory interaction”. Although the aforementioned core inhibitory interaction remains intact in Vav1 Δ 65, the truncated form Vav1 is more competent in cell transforming activity than the full-length wild type Vav, suggesting that the core interaction cannot account for all the inhibitory regulation originated from the N-terminal CH domain and Ac region. There probably exist

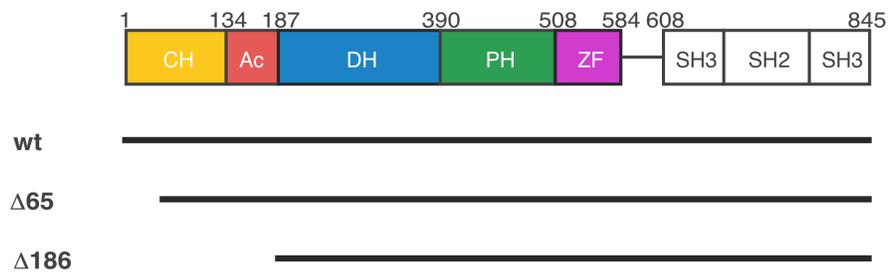


Figure 1-2. Different truncations of Vav. Vav was first isolated as a form with the first 65 residues truncated. $\Delta 65$ exhibits cell transforming activity upon over-expression. Co-expression of tyrosine kinase, Lck, further enhances the transforming activity. Further truncation to eliminate the entire CH and Ac results in more potent transforming activity, which is phosphorylation-independent.

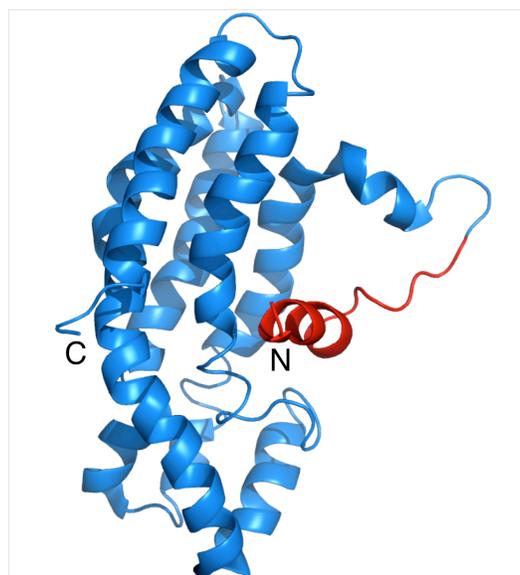


Figure 1-3. Structural basis of Vav1 autoinhibition. A helix derived from the acidic region binds to the active surface of the Dbl homology (DH) domain and thereby inhibits DH activity. The Dbl homology (DH) is colored in blue and the helix is colored in red.

other domain-domain contacts contributing the autoinhibition. The inferred inhibitory intramolecular interactions, while of unknown structural origin until recently, are referred to as regulatory interactions. How the regulatory interaction further suppresses Vav activity *in vivo* was unclear prior to the work described below.

Vav receives a variety of signal inputs emanating from membrane receptors via different domains to downstream effectors. A number of post-translational modifications, for example, tyrosine phosphorylation (Deckert, Tartare-Deckert et al. 1996; Pisegna, Zingoni et al. 2002; Tamas, Solti et al. 2003), tyrosine dephosphorylation (Stebbins, Watzl et al. 2003; Wu, Katrekar et al. 2006), arginine methylation (Blanchet, Cardona et al. 2005), mono-ubiquitination (Rao, Dodge et al. 2002; Miura-Shimura, Duan et al. 2003; Simmons, Gangadharan et al. 2005) and caspase-dependent degradation (Hofmann, Hehner et al. 2000), can occur depending upon upstream signaling stimuli. Tyrosine phosphorylation is the best-established post-translational modification of Vav (Bustelo and Barbacid 1992; Bustelo, Ledbetter et al. 1992; Amarasinghe and Rosen 2005). The Ac domain is phosphorylated in response to a variety of ligand-stimulated receptors, including B- and T-cell receptors through the action of Syk and Src-family tyrosine kinases (Bustelo and Barbacid 1992; Crespo, Schuebel et al. 1997; Han, Das et al. 1997; Movilla and Bustelo 1999; Lopez-

Lago, Lee et al. 2000). Phosphorylation is associated with increased guanine nucleotide exchange (GEF) activity (Crespo, Schuebel et al. 1997; Abe, Whitehead et al. 1999; Mosteller, Han et al. 2000) and stimulation of downstream signals such as Rac activation, actin assembly and transcriptional activation of a number of important genes involved in hematopoietic cell development. There are three sites of tyrosine phosphorylation in the Ac region, Tyr142, Tyr160 and Tyr174, of which Tyr174 is the most important modification site and the most extensively studied (Bustelo and Barbacid 1992; Bustelo, Ledbetter et al. 1992; Aghazadeh, Lowry et al. 2000; Amarasinghe and Rosen 2005). Phosphorylation or phosphomimicking mutation of Y174 enhances GEF activity *in vitro*. Taken together, these data imply that the N-terminus of Vav is autoinhibitory toward the DH domain GEF activity, and phosphorylation of Tyr174 relieves this negative regulation. In summary, cell surface receptors communicate with Vav via Syk- and Src-family tyrosine kinase dependent phosphorylation.

Through different domains, numerous protein-protein and protein-lipid interactions also occur to either regulate Vav activity or to execute GEF-dependent and independent Vav function. Calmodulin (Zhou, Yin et al. 2007), APS (Yabana and Shibuya 2002), Ly-GDI (Groysman, Russek et al. 2000), EZH2 (Nolz, Gomez et al. 2005; Su, Dobenecker et al. 2005) and ENX-1 (Hobert, Jallal et al. 1996) have been shown to associate with the CH domain of Vav. Socs1 (De Sepulveda, Okkenhaug et al. 1999) and SHP-1 (Jones, Craik et al. 2004) have

been found to associate with the acidic region of Vav in phosphorylation-independent and phosphorylation-dependent fashion, respectively. It has been demonstrated that the phosphatidylinositol 3-kinase substrate phosphatidylinositol-4,5-bisphosphate inhibited activation of Vav by the tyrosine kinase Lck, whereas the product phosphatidylinositol-3,4,5-trisphosphate enhanced phosphorylation and activation of Vav by Lck (Han, Luby-Phelps et al. 1998; Das, Shu et al. 2000). There are numerous interactions involving the SH3-SH2-SH3 modules in the C-terminus of Vav (Wu, Motto et al. 1996; Bustelo 2001; Billadeau, Upshaw et al. 2003). These interactions are proposed to be responsible for proper localization of Vav (Bustelo 2001). It will not be surprising, however, if the SH3-SH2-SH3 module functions beyond localization. Although not demonstrated, it is conceivable that some of these interactions modulate Vav by modulating some of the intramolecular interactions.

Pre-equilibrium view of autoinhibition and its role in activation

Activities of signaling molecules require tight temporal and spatial regulation to ensure signal fidelity. Deregulation of signal often leads to diseases such as cancer. One class of molecules are suppressed via intramolecular interactions, i.e. autoinhibition (Xu, Harrison et al. 1997; Aghazadeh, Lowry et al. 2000; Pufall and Graves 2002). Autoinhibited systems normally require non-covalent interactions and/or covalent post-translational modifications for activation (Abdul-Manan, Aghazadeh et al. 1999; Aghazadeh, Lowry et al. 2000;

Kim, Kakalis et al. 2000; Pufall and Graves 2002; Cheng, Skehan et al. 2008). For example, activation of the autoinhibited $\sigma 70$ prokaryotic transcriptional factor requires intermolecular interactions with RNA polymerase (Dombroski, Walter et al. 1992; Pufall and Graves 2002); activation of autoinhibited Src requires external ligands for the SH2 domain and the SH3 domain (Alexandropoulos and Baltimore 1996; Pufall and Graves 2002); phosphorylation of the inhibitory helix of Vav is required for its activation (Aghazadeh, Lowry et al. 2000); Cdc42 binding to GBD domain is needed to activate WASP (Abdul-Manan, Aghazadeh et al. 1999; Kim, Kakalis et al. 2000; Leung and Rosen 2005). Often such modification sites are inaccessible to modifying enzymes in the ground state structures (Aghazadeh, Lowry et al. 2000) or the autoinhibited structures are incompatible with incoming binding partners (Abdul-Manan, Aghazadeh et al. 1999; Aghazadeh, Lowry et al. 2000; Kim, Kakalis et al. 2000; Cheng, Skehan et al. 2008). Such observations suggest a puzzle about how the seemingly inaccessible sites are eventually accessed. This is a feature of most allosteric systems in general. For example, while the substrate binding pocket of HIV protease (Rick, Erickson et al. 1998) and the hydrophobic ligand binding cavity of a mutant T4 lysozyme (Eriksson, Baase et al. 1992) are inaccessible in their ground state structures, binding of these ligand/substrate is actually very rapid. A pre-equilibrium model is often proposed to reconcile this conundrum (Monod, Wyman et al. 1965; Mulder, Mittermaier et al. 2001; Volkman, Lipson et al. 2001;

Leung and Rosen 2005; Li, Martins et al. 2008). The model 1) invokes a pre-equilibrium in the protein between a ground state in which the sites are buried and a high-energy excited state in which the previously inaccessible sites are exposed and 2) modifications simply stabilize the excited state(s) versus the ground state and therefore activate the proteins. Following this notion, the model also predicts that the excited states should resemble the activated forms of the once-autoinhibited proteins. A quantitative prediction stemming from this model is that if fluctuations are fast, the populations of ground and excited states should correlate with biochemical properties; if fluctuations are slow, their kinetics should correlate with biochemistry. This model is extremely difficult to fully verify, especially in quantitative rigor, due to the elusive nature of the excited state and often instability of the activated form of the proteins. Difficulties can also arise because of complexity of dynamics (e.g. greater than two states in exchange), and because proper analysis requires quantitative correlations between dynamics and biochemistry. Nevertheless this pre-equilibrium proposition is a potential operational mode that is operating in multi-domain autoinhibited signaling molecules like Vav and Src. Part of chapter 2 is devoted to dissect out and depict this operational mode in quantitative rigor in AD.

Cooperative autoinhibition

One feature of Vav autoinhibition is its hierarchic/cooperative construction, i.e. multiple weak interactions collectively provide stronger

inhibition necessary for biology. However, the hierarchical autoinhibition structure is not unique to Vav. N-terminal inhibitory helices are also present in several other Dbl family GEFs including Tim (Yohe, Rossman et al. 2007), Ngef (Yohe, Rossman et al. 2008) and Wgef (Yohe, Rossman et al. 2008). Truncation of the inhibitory helix, phosphorylation or phosphomimicking mutations of these GEFs lead to their activation just like Vav. Interestingly all of these GEFs possess a SH3 domain at the C-terminus, which binds *in cis* to a polyproline peptide between the inhibitory helix and the DH domain and negatively regulate their GEF activities (Yohe, Rossman et al. 2007; Yohe, Rossman et al. 2008). Most likely, as depicted in Figure 1-4, the inhibitory helix physically blocks the GEF activity bearing domains and extra interactions between the SH3 domain and polyproline motif strengthen the former interaction (Yohe, Rossman et al. 2007; Yohe, Rossman et al. 2008). As in Vav, the former can be considered as the core inhibitory interactions and the latter are then called the regulatory interactions. The physical basis of how the regulatory interactions strengthen the core interactions is not understood.

In the Grp1 family of guanine nucleotide exchange factors for the Arf GTPase, the linker between the GEF domain and the PH domain and a C-terminal polybasic region bind to the GEF domain mimicking the substrate, Arf (DiNitto, Delprato et al. 2007). The two interactions act synergistically in suppressing GEF

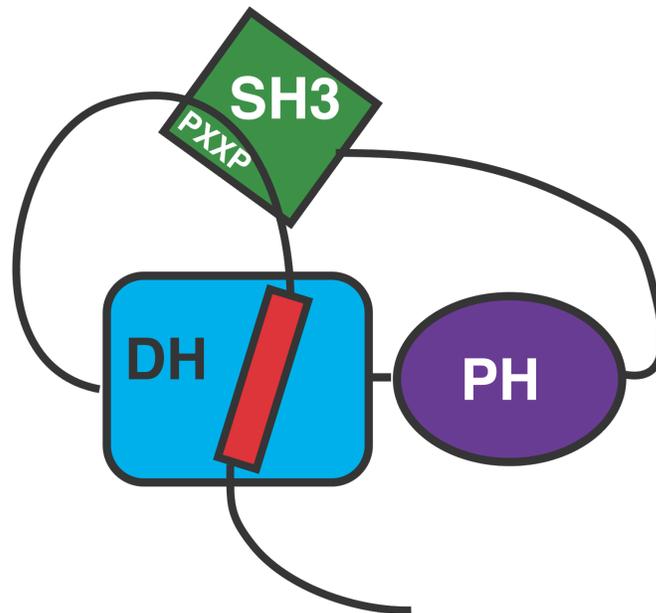


Figure 1-4. Structural models for Tim, Ngef and Wgef. A predicted helix binds to the DH domain of these Dbl family GEFs and therefore inhibits DH domain activity. C-terminal SH3 domain binds the poly-proline motif in the linker region between the inhibitory helix and the DH domain.

activity by stabilizing the autoinhibited conformation (DiNitto, Delprato et al. 2007). In SH-PTP2, a protein tyrosine phosphatase, there are two tandem SH2 domains and both involve in autoinhibition of SH-PTP2 phosphatase activity. Displacement of individual SH2 domain results in c.a. 10-fold increase in phosphatase activity and displacement of both domains results in 37-fold increase, suggesting that these two SH2 domains cooperatively inhibit SH-PTP2 activity (Pluskey, Wandless et al. 1995). In another class of well-known multi-domain signaling proteins, Src family kinases, inhibition of the kinase domain is mediated by both interactions between the SH2 and a C-terminal phosphotyrosine motif and interactions between the SH3 and a polyproline motif flanked by the SH2 domain and the kinase domain (Xu, Harrison et al. 1997; Xu, Doshi et al. 1999). Disruption of either interaction leads to elevated kinase activity (Briggs and Smithgall 1999). A polypeptide containing both SH2 and SH3 ligands is a synergistic activator of Src (Alexandropoulos and Baltimore 1996). In Abl, a nonreceptor tyrosine kinase related to Src, the SH3 interaction is conserved. While the corresponding SH2-tail interaction is not observed, the SH2 domain associates with the C-lobe of the kinase domain. In addition to these two interactions, unexpectedly an N-terminal myristoyl moiety wraps around the protein and plugs deeply into the C-lobe of the kinase domain. All three interactions cooperatively inhibit Abl kinase activity (Hantschel, Nagar et al. 2003; Nagar, Hantschel et al. 2003).

The list of molecules possessing more than one inhibitory interaction will surely be lengthening with time. In them, the intramolecular contacts offer moderate inhibition individually but collectively achieve high degrees of inhibition. The construction of using multiple weak interactions in hierarchical autoinhibition is advantageous over the construction of single strong interactions since 1) the former is potentially faster in response to upstream cues than the latter; 2) the former has kinetic proofreading potential and is therefore less susceptible to random cellular noise (Weiss and Nilsson 2003) and 3) the former can accommodate multiple signal inputs, further ensuring signal fidelity. Due to the importance of molecules of this construction, many structural studies have been conducted on such molecules (Xu, Harrison et al. 1997; Xu, Doshi et al. 1999). Despite this structural information, the energetic mechanisms by which individual domains modulate each other's biochemical activity are largely unknown. The second part of chapter 2 is to provide a thermodynamic basis to the cooperative autoinhibition by extending our understanding of AD to the full regulatory element of the protein (CADPZ) (Fig. 1-1). In this way, I learn how the protein simultaneously meets the cellular demands for high-level repression in the inactive state and rapid activation in response to stimulating signals.

In hierarchic/cooperative autoinhibited systems, one interesting question is what is the strength of energetic coupling between individual interactions. The coupling strength is essentially the maximum information flow capacity in these

systems via allostery. Monod, Wyman, and Changeux provided a thermodynamic foundation for explaining coupling strength, referred to as MWC model (Monod, Wyman et al. 1965). Although this model was developed in the context of protein-ligand interactions, it provides a universal framework for depicting the energetic coupling of conformational changes between two processes. Central to the concept of allostery is the assumption of a pre-equilibrium between two conformations, the T and R states. If ligand has different affinities towards either the T or R state (i.e. $K_T \neq K_R$, in which K_T and K_R are the association constants of the ligand to the T and R states, respectively), ligand binding will bias the pre-equilibrium towards the favorable, i.e. high affinity, state. A term, the C parameter, is defined as the ratio of K_R and K_T and describes the coupling strength of the ligand binding towards the pre-equilibrium.

The C parameter is normally characterized indirectly, for example by measuring enzymatic activity change before and after ligand binding (Leung and Rosen 2005). However, this conventional method cannot be used to characterize the C parameter in systems where enzymatic activity is unavailable. Energetic coupling is a versatile way of information flow within proteins regardless of whether there is enzymatic activity handy for assaying. In addition to energetic coupling, i.e. thermodynamic coupling, kinetic coupling is an integral part of allostery. In some systems, the kinetics of conformational changes are really the rate-limiting element. For example, the N-terminal part of syntaxin inhibits its C-

terminal part from incorporating into the four-helix bundle SNARE complex via impairing assembly kinetics but not thermodynamics (Nicholson, Munson et al. 1998). Kinetic coupling is not addressed in MWC model, since it is based on thermodynamics. In fact, kinetic coupling has been largely ignored due to technical challenges. In Chapters 3 and 4, both thermodynamic and kinetic coupling strengths of the two dynamic processes in AD are characterized via direct parameterization of a four-state equilibrium derived from the processes using NMR CPMG techniques detailed below.

Experimental approaches to study protein dynamics

Mature X-ray crystallographic, NMR spectroscopic methods, joined now by developing electron microscopy offer enormous information about how proteins work by providing average low energy atomic structures. A complete picture of protein function at atomic resolution requires both high-resolution structure(s) of kinetically stable state(s) (defined in an influential paper by Frauenfelder et. al (Frauenfelder, Sligar et al. 1991)) and time evolution of atomic coordinates, the latter of which is dynamics. The aforementioned techniques also provide a plethora of evidence of dynamics. The B-factor in X-ray structures is an index of local flexibility to some degree (Palmer 2001). In addition, some X-ray crystallographic studies reveal different conformations within the same crystal, each of which corresponds to a different functional form of the protein. For example Matthew and Faber demonstrated that a mutant T4 lysozyme can adopt

five different conformations (Faber and Matthews 1990). In another example, Kern and coworkers have discovered three molecules in an asymmetric unit of apo Aquifex adenylate kinase, each of which is of distinct structure to the others (Henzler-Wildman, Thai et al. 2007). That these conformations of T4 lysozyme or adenylate kinase can be observed in crystalline condition suggests that the proteins sample these states in solution. Unfortunately, dynamics information from these studies relies heavily on luck in obtaining such crystal forms. Of course, conformational snapshots of proteins can also be routinely obtained in crystallographic studies using ligand, substrate, inhibitor and mutagenesis (Taylor, Yang et al. 2004). But the essence of protein dynamics--kinetics and thermodynamics--is completely inaccessible in techniques like X-ray crystallography. In protein NMR spectrum there are more than one set of resonances derived from the same moieties if it has two or more slowly interconverting states. In such cases, a qualitative sense of thermodynamics (populations of states) can be inferred from relative peak intensities. Dynamic information from such NMR studies is at the mercy of slow interconverting rates and high enough populations for sufficient signal/noise ratio for detection.

Several types of NMR methods have been developed to analyze protein dynamics from picoseconds to seconds (Fig. 1-5). Spin relaxation measurements (Palmer 2001) NMR residual dipolar coupling (RDC) (Bax and Grishaev 2005; Lange, Lakomek et al. 2008), spin relaxation rate constant in rotating frame

(R1ρ) (Palmer and Massi 2006) and heteronuclear NMR exchange (ZZ-exchange) spectroscopy (Palmer, Kroenke et al. 2001) have been developed to study ps-ns, ns-μs, μs, subseconds to seconds, respectively (Fig. 1-5). NMR dynamics techniques based on the Carr-Purcell-Meiboom-Gill (CPMG) relaxation dispersion are especially useful since 1) they can provide both kinetic and thermodynamic information of chemical exchanges, 2) they can be used to study chemical exchange between populations as skewed as 99.5/0.5 and 3) there are more biologically relevant motions in its responsive time scale, μs-ms, than at other time scales (Palmer, Kroenke et al. 2001). Palmer and coworkers reported the first widely used CPMG pulse sequence based on single quantum (SQ) ^{15}N relaxation in ^1H - ^{15}N amide moieties (Loria, Rance et al. 1999). To increase detection sensitivity, both of the Palmer and Kay groups took the advantage of transverse relaxation optimized spectroscopy (TROSY), developing TROSY versions of SQ ^{15}N CPMG pulse sequences shortly after (Loria, Rance et al. 1999; Tollinger, Skrynnikov et al. 2001). This experiment was extended to SQ of sidechain NH_2 (Mulder, Skrynnikov et al. 2001). To provide more orthogonal measurements with few new fitted parameters, Kay and coworkers also developed five complementary CPMG experiments of ^1H - ^{15}N amide, SQ ^1H , multiple quantum (MQ) of ^1H - ^{15}N , MQ of ^{15}N - ^1H , zero quantum (ZQ) of ^1H - ^{15}N , double quantum (DQ) of ^1H - ^{15}N (Korzhnev, Neudecker et al. 2005).

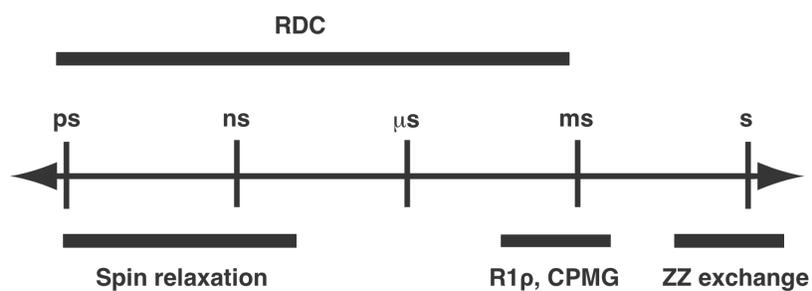


Figure 1-5. Time scales for NMR dynamics techniques. Spin relaxation measurements are sensitive to picoseconds-nanosecond time scale motion. Spin relaxation measurements are sensitive to microseconds time scale motions. Carr-Purcell-Meiboom-Gill technique is sensitive in microsecond-millisecond time scale motions. ZZ exchange is suitable for subsecond to second motions. Residual dipolar coupling covers motions from picoseconds - milliseconds.

Soon after the amide ^{15}N CPMG pulse sequences were developed, analogous sequences to measure SQ (Skrynnikov, Mulder et al. 2001; Korzhnev, Kloiber et al. 2004; Lundstrom, Vallurupalli et al. 2007) and MQ (Korzhnev, Kloiber et al. 2004) methyl ^{13}C relaxation were developed to take the advantage of slow relaxation (and high signal/noise) of methyl groups with regards to amide groups. CPMG experiments based on the amide ^{15}N nucleus are normally used in molecules smaller than 25 kDa (Boehr, Dyson et al. 2006). Due to the more favorable relaxation behavior of methyl groups, methyl ^{13}C based CPMG techniques have been extended to systems as large as 82 kDa (Korzhnev, Kloiber et al. 2004) and amazingly 300 kDa (Sprangers, Gribun et al. 2005). All these CPMG experiments are based on measurement of relaxation dispersion, the dependence of transverse relaxation rate, R_2 , on the strength of an applied transverse magnetic field, of certain coherences (SQ, DQ, MQ and ZQ) of certain moieties (amide or methyl). In a system that fluctuates in equilibrium between two states, the populations of the states, the rates of interconversion, and the difference in chemical shifts between them can be determined from analyses of relaxation dispersion.

Functional Relevance of Microsecond-millisecond Dynamics

Protein dynamics in microsecond-millisecond (μs - ms) scales play important roles in many aspects of protein function since there are many functional processes happening in this time scale (Kay 1998; Palmer, Kroenke et

al. 2001; Palmer 2004; Kay 2005; Kern, Eisenmesser et al. 2005; Palmer, Grey et al. 2005; Boehr, Dyson et al. 2006). For example enzymatic k_{cat} values are mainly between 10^3 and 10^6 s^{-1} (Wolfenden and Snider 2001); protein folding rates range from $0.2 \sim 10^5 \text{ s}^{-1}$ (Palmer, Kroenke et al. 2001), protein-protein interaction or small molecule ligand-protein recognition often happen in microseconds or slower and allosteric regulation switches in milliseconds or slower (Palmer, Kroenke et al. 2001). Elucidating the relationship between dynamics and function is vital in understanding allostery, enzyme catalysis, protein engineering and drug design (Boehr, Dyson et al. 2006; Goodey and Benkovic 2008). NMR dynamics techniques play a major role in providing such pivotal information by demonstrating quantitative correspondence between dynamic parameters and biochemical parameters (Kay 1998; Wand 2001; Boehr, McElheny et al. 2006; Henzler-Wildman and Kern 2007; Loria, Berlow et al. 2008).

Qualitative correlations between dynamics and function

Some of the first work used to correlate NMR dynamics to function was qualitative in nature, and involved studies of bacterial two-component signaling modules. In the spin relaxation measurement, the presence of R_{ex} suggests the existence of μs - ms dynamics, which often triggers the temptation to speculate relationships between slow scale motions and function. Spo0F, a *Bacillus subtilis* response regulator, is crucial in the two-component signaling pathway that

regulates the morphological change during vegetative state to dormant state transition (Burbulys, Trach et al. 1991). Spo0F physically associates with its effector proteins via a semi-contiguous surface upon activation by phosphorylation (Tzeng and Hoch 1997; Tzeng, Feher et al. 1998). In a study of the internal dynamics of Spo0F, Feher and Cavanagh discovered that the regions showing significant R_{ex} coincide with the surface that is critical for protein-protein interactions (Feher and Cavanagh 1999). Nitrogen regulatory protein C (NtrC) is another well-known member of two-component systems in bacteria. Phosphorylation of a conserved aspartate residue in the receiver domain of NtrC results in a large conformational change required for downstream function (Rombel, North et al. 1998). In a similar study on the receiver domain of NtrC, Kern and coworkers analyzed the internal dynamics of this related protein (Volkman, Lipson et al. 2001). They also discovered that a region experiencing μ s-ms motions coincides with the region of structural changes upon activation by phosphorylation. Importantly, such motions are quenched upon phosphorylation (Volkman, Lipson et al. 2001). In both studies, the authors speculated that there exists a pre-equilibrium between an inactive conformation and an active conformation in the apo proteins and phosphorylation merely redistributes the population by favoring the active conformation.

Pre-equilibria in binding and catalysis

The proposition of a pre-equilibrium view of Spo0F and NtrC receiver

domains and its relevance in function using spin-relaxation measurement is of somewhat speculative nature. Nevertheless, these studies are the first to suggest the existence of pre-equilibrium using NMR dynamics techniques. Some recent demonstrations on this view are more definitive. *Escherichia coli* dihydrofolate reductase (DHFR) catalyzes the reduction reaction of 7,8-dihydrofolate (DHF) to 5,6,7,8-tetrahydrofolate (THF) using reduced nicotinamide adenine dinucleotide phosphate (NADPH) as a cofactor (Fierke, Johnson et al. 1987). In one catalytic cycle, DHFR steps through five intermediates: E:NADPH, E:NADPH:DHF, E:NADP⁺:THF, E:THF, E:THF:NADPH (Fierke, Johnson et al. 1987). In a recent study using CPMG approaches, Wright and coworkers elegantly demonstrated that each intermediate experiences low-populated excited states whose structures resemble the native structures of the preceding and following intermediates (Boehr, McElheny et al. 2006). This set the stage for subsequent events (substrate or cofactor binding, product release, covalent bond breakage or formation) to modulate the pre-equilibrium such that the current excited state becomes the native state of the next intermediate. In an equally dramatic case, using RDC measurements in a number of solution conditions, Lange *et al.* (Lange, Lakomek et al. 2008) recently determined a structural ensemble for ubiquitin that depicts the dynamic landscape in ns- μ s time scale. Amazingly, all of the 46 known ubiquitin structures in various complexes fall within the ensemble of structures of the free protein (Lange, Lakomek et al. 2008). This result echoes the

studies above and argues strongly that proteins are dynamic ensembles (Yon, Perahia et al. 1998). An interesting question is why there are up to hundreds of conformations with similar energies (which are thus all reasonably populated) in ubiquitin and only a handful of such conformations in Spo0F, NtrC receiver domain, DHFR and many other proteins to be discussed below. Does the number of conformations correlate with the versatility of the proteins, i.e. ubiquitin is involved in an enormous number of pathways and has to interact with many different proteins while Spo0F and NtrC only interact with limited number of proteins and each intermediate of DHFR only needs to be prepared for the next catalytic step?

Quantitative correlations between dynamics and function

CPMG measurements can be used to reveal quantitative relationships between dynamics and function, especially in ligand binding reactions and in enzymatic activity. For instance, there is a cavity in the X-ray structure of a mutant form of phage T4 lysozyme, L99A, which is sterically hindered from solvent access (Eriksson, Baase et al. 1992). However, small hydrophobic aromatic compounds like benzene and xenon can rapidly bind within the cavity (Eriksson, Baase et al. 1992). Using a variety of CPMG techniques, Kay and coworkers discovered an excited state in L99A, in which residues around the cavity are involved in a concerted motion (Mulder, Mittermaier et al. 2001; Mulder, Skrynnikov et al. 2001; Skrynnikov, Mulder et al. 2001; Korzhnev,

Orekhov et al. 2003). In addition the ligand binding rate characterized using orthogonal approaches is similar as the chemical exchange rate determined via CPMG measurement (Mulder, Mittermaier et al. 2001). This correlation suggested that these motions are responsible for exposing the cavity for small molecule ligands and that the opening rate is the rate-limiting step of ligand binding. These elegant studies on T4 lysozyme demonstrated the promise to further elucidate structure-dynamics-function relationships in a variety of proteins by extending these techniques to other ligand binding reactions or other proteins including enzymes (Boehr, Dyson et al. 2006).

It has been suggested that enzymes have evolved such that the chemical reactions *per se* are very fast in nature and the rate limiting steps of enzyme catalysis are the intrinsic conformational changes necessary for steps like substrate or cofactor binding and product release (Wolfenden and Snider 2001; Loria, Berlow et al. 2008). In an early study conducted on ribonuclease A (RNase A), Loria and coworkers demonstrated a concerted conformational change experienced by both the active site and other substrate binding sites (Cole and Loria 2002). In a following study, they showed that the conformational change observed in the apo-RNase A is preserved in a substrate-mimic bound form (Beach, Cole et al. 2005). In addition, the motions in the two forms of proteins sample the same two end states corresponding to the open and closed form of the enzyme. The open and closed forms are more populated in the apo-enzyme and

ligand-saturated enzyme, respectively. These results further substantiate the notion that a preexisting dynamic equilibrium in the apo-protein samples an excited state resembling the substrate-saturated ground state and that substrate binding simply stabilizes the bound conformer. More importantly, the exchange rate of the conformational change in bound enzyme coincides with the k_{cat} value, suggesting a potential link between the motion and catalysis (Beach, Cole et al. 2005). They have revealed that the product release rate, k_{off} , also coincides with both k_{ex} and k_{cat} (Kovrigin and Loria 2006). In addition, all three quantities also have the same kinetic solvent isotope-effect (KSIE) suggesting they are coupled with each other (Watt, Shimada et al. 2007). Mutation of a conserved aspartate 121 to alanine modulates all three quantities to the same extent (Kovrigin and Loria 2006). These results strongly suggest that the conformational change is linked with product release, which is the rate-limiting step in the enzymatic catalysis (Kovrigin and Loria 2006). In a recent study, they have identified a single residue, His48, as being responsible for the link between conformational change and the product release as H48A mutation diminishes the conformation change and the KSIE of k_{cat} (Watt, Shimada et al. 2007). These studies collectively revealed that the μs - ms conformational change indeed dictates the catalytic power of RNase A.

In a parallel series of studies, Kern and coworkers have studied μs - ms dynamics in cyclophilin A (CypA), an enzyme catalyzing peptidyl prolyl *cis/trans*

isomerization, using a number of NMR techniques (Eisenmesser, Bosco et al. 2002; Kern and Zuiderweg 2003; Eisenmesser, Millet et al. 2005; Kern, Eisenmesser et al. 2005). Motions detected during catalysis for both backbone ^{15}N and sidechain ^{13}C probes could be globally fitted to a single k_{ex} of ca. 2700 s^{-1} . As in RNase A, the k_{ex} value correlates very well with the turnover rate of peptidylprolyl *cis/trans* isomerization suggesting a potentially dictating role of the observed motion in catalysis (Eisenmesser, Millet et al. 2005). Similar relationships have also been suggested in the ribonuclease binase (Wang, Pang et al. 2001), DHFR (Boehr, McElheny et al. 2006) and adenylate kinase (Wolf-Watz, Thai et al. 2004).

In summary, there are quite a few examples in which μs - ms dynamics have been identified and linked to function. More work is needed to assess how general observed correlations between dynamics and function are. In addition, most of the existing studies assigned the functional relevance based on single data point, for example, similar exchange rates in internal motion with enzyme turnover (Eisenmesser, Bosco et al. 2002) or ligand binding (Mulder, Mittermaier et al. 2001). More rigorous correlations based on a series of data points derived from different perturbations are yet to be demonstrated. It is also worth noting that the vast majority of quantitative analyses have been done on catalysis, and that such work has not yet been done on regulatory interactions.

Co-existing Multiple Dynamic Processes

As CPMG and other NMR methods have been more broadly applied, evidence is emerging that some (perhaps many) proteins have multiple dynamic processes in the μs -ms time regime. There is a plethora of reports in which dynamic landscapes beyond two-state are required to better account for peptide-protein binding reactions (Hensmann, Booker et al. 1994; Gunther, Mittag et al. 2002), inhibitor-enzyme interactions (Perlman, Davis et al. 1994; Curto, Moseley et al. 1996; Deng, Zhadin et al. 2001; Gulotta, Deng et al. 2002; Hammes 2002) or single proteins (Eisenmesser, Bosco et al. 2002; Li, Martins et al. 2008; Masterson, Mascioni et al. 2008). Quantitative analyses of such complex systems is much harder since the number of parameters increases and correlations between parameters in the fitting cannot be resolved. The first quantitative study was achieved via CPMG measurements (Tolkatchev, Xu et al. 2003). In an analysis of CPMG data of an antithrombin peptide binding to a small fraction of human prothrombin (< 3.5%), Ni and coworkers have discovered abnormal or inconsistent fitted parameters for some resonances if they assume the two-state model holds (Tolkatchev, Xu et al. 2003). These discrepancies can be explained by the presence of two exchanging conformations for the binary complex. The three-state model (Fig. 1-6c) was solved quantitatively by global fitting data of three resonances at two static fields and three prothrombin concentrations (Tolkatchev, Xu et al. 2003). Using NMR titrations, Wright and coworkers

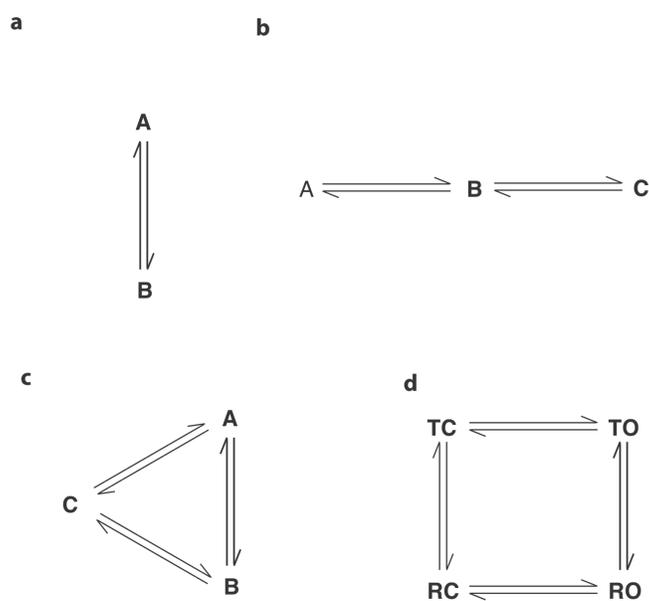


Figure 1-6. Thermodynamic models for two- (a), linear three-state (b), circular three-state (c) and four-state (d). The nomenclatures for the states in (a-c) are arbitrary. The nomenclatures in (d) are composed of two sets of binary letters, T/R and C/O, to depict the fact that the four states are derived from binary combination of two processes, TR process and CO process.

recently found that the binding reaction between the phosphorylated kinase inducible activation domain (pKID) of the transcription factor CREB and the KIX domain of the CREB binding protein is in the slow exchange regime, i.e. resonances corresponding to both the free and complexed form of pKID are observed in partially saturated pKID samples (Boehr, McElheny et al. 2006). However, some resonances show fast exchange behavior towards a direction deviating from the bound state during the process of titrating KIX into pKID. This observation suggests an in-pathway intermediate along the binding reaction trajectory (Fig. 1-6b). Wright and coworkers not only uncovered the nature of the intermediate via complementary studies but also solved the linear three-state model using CPMG relaxation dispersion data for several resonances at two static fields and four KIX concentrations (Boehr, McElheny et al. 2006). It turns out that the intermediate is an ensemble of encounter-complexes along the binding trajectory.

Multiple dynamic processes are also observed in single proteins including single domain proteins. Palmer and coworkers have observed two dynamic processes existing in the basic pancreatic trypsin inhibitor (BPTI) based on a biphasic behavior in ^{15}N relaxation dispersion studies at certain temperatures for some resonances (Grey, Wang et al. 2003). Kinetics of the two processes in BPTI differ by more than 10-fold. CPMG data of four resonances at two static fields and three temperatures were used to solve this linear three-state system (Grey,

Wang et al. 2003). Kay and colleagues also identified a linear three-state equilibrium in the folding-unfolding process of the SH3 domain of Fyn, i.e. a folded-intermediate-unfolded three-state equilibrium (Fig. 1-6b) (Korzhnev, Salvatella et al. 2004). In an early study, CPMG measurements of a single protein at multiple fields and multiple temperatures were successfully used to solve the linear three-state model (Korzhnev, Salvatella et al. 2004). Later, a suite of six CPMG experiments was applied to a single protein at a single temperature enabling the three-state equilibrium to be solved more robustly (Korzhnev, Neudecker et al. 2005; Neudecker, Korzhnev et al. 2006). According to Eisenmesser et al. in addition to fluctuations in the protein core, CypA also has a loop with motions at a different rate (Eisenmesser, Millet et al. 2005). The two processes (the core fluctuations and the loop motions) appear to be coupled, since mutations preferentially affecting the core process also perturb the loop motion and *vice versa* (Eisenmesser, Millet et al. 2005). Substrate peptides in both *cis*- and *trans*-configurations are capable of binding with CypA and substrate bound enzyme has a process corresponding to *cis/trans* isomerization (Eisenmesser, Bosco et al. 2002). Therefore, there are at least five processes that exist in CypA in the presence of partially saturating amount of substrate peptide: two in free enzyme, one in bound enzyme and two involving substrate binding. The coexistence of multiple processes therefore seems to be a prevalent phenomenon.

A number of the abovementioned studies have demonstrated that it is feasible to solve a linear or a circular three-state model if either the kinetics of two processes differ over 10-fold (Grey, Wang et al. 2003), multiple perturbations (e.g. ligand concentrations (Tolkatchev, Xu et al. 2003), temperatures (Korzhnev, Salvatella et al. 2004), pressures) are possible or multiple orthogonal NMR measurements are available (Korzhnev, Neudecker et al. 2005).

One common scenario for multiple and coexisting dynamic processes is that one process pre-exists in a protein and another process, e.g. ligand binding, *in cis* protein-protein interaction, *in trans* protein-protein interactions or another intrinsic process in the same domain, superimpose onto the former. As will be detailed in Chapters 2-4, two processes were discovered in AD (Li, Martins et al. 2008). The two processes are thermodynamically and kinetically coupled such that changing the population distribution in one also changes the population distribution and kinetics in the other. The two dynamic processes in CypA serve as a perfect example in which two intrinsic processes co-exist (Eisenmesser, Bosco et al. 2002; Eisenmesser, Millet et al. 2005). Assuming each process has two states, the binary combination of the two processes gives rise to a four-state equilibrium (Fig. 1-6d). The fact that mutations in one process in CypA also perturb the other process also suggests these processes are energetically coupled (Eisenmesser, Millet et al. 2005). Coupling between equilibria appears to be a central feature of regulation in proteins.

Despite the increasing recognition that many proteins are likely to possess multiple coupled motions on the μ s-ms timescale, there is a poor understanding of how dynamic processes interact with one another. What is the magnitude of coupling? How is the interaction between dynamic processes related to function? The absence of such information stems in part from the fact that methods have not been reported for quantitative analysis of coupled equilibria. Moreover, there is little understanding of what the practical limits of such analyses are likely to be, and how biochemical and NMR methods can be combined to extend these in particular cases. The work in Chapter 3 seeks to explore these issues *in silico* through analysis of the four-state equilibrium (Fig. 1-6d) and to provide an approach (and understanding of the limits thereof) for quantitative analyses of four-state equilibria in general. The work in Chapter 4 describes work to quantitatively understand the thermodynamic coupling in the Vav1 AD element and thus the energetic construction of the core autoinhibited element in Vav1.

Chapter 2 Internal dynamic control activation and activity of the autoinhibited Vav DH domain¹

Introductions

Proteins are dynamic entities that transition among numerous conformational states that are important for stability, regulation, and activity (Palmer, Kroenke et al. 2001; Kern and Zuiderweg 2003; Kay 2005). Autoinhibited proteins are striking examples of the interplay between function and dynamics, since these molecules typically require large structural changes to switch between inactive and active states. Autoinhibition is normally achieved through the binding of a regulatory element to an activity-bearing domain located in a distinct region of the same polypeptide chain. These interactions often block binding and/or catalytic activity by directly occluding the active site (Pufall and Graves 2002; Dueber, Yeh et al. 2003). Relief of autoinhibition typically requires activators to bind and/or covalently modify sites in the regulatory element that are buried in the autoinhibited structure. Internal dynamics that provide access to such sites likely play important roles in regulation and activation of autoinhibited

¹ This chapter is adapted, with limited modifications from my recently published paper (P. Li, P., I. R. Martins, et al. (2008). "Internal dynamics control activation and activity of the autoinhibited Vav DH domain." Nat Struct Mol Biol **15**(6): 613-8.

systems. However, this hypothesis has not yet been tested through quantitative correlations between dynamics and function.

The proto-oncogene product Vav is an autoinhibited protein that relays signals from numerous cell surface receptors, such as the T cell, B cell and Fc γ receptors, to intracellular pathways that control cytoskeletal structure and transcription (Bustelo 2001; Turner and Billadeau 2002; Tybulewicz 2005). Vav is a multi-domain protein that contains a Dbl homology (DH) domain, which functions as a guanine nucleotide exchange factor (GEF) for small GTPases in the Rho family (Abe, Whitehead et al. 1999; Heo, Thapar et al. 2005). The GEF activity of Vav is negatively regulated in part by binding of the DH domain to an adjacent element termed the acidic region (Ac) (Abe, Whitehead et al. 1999; Movilla and Bustelo 1999; Aghazadeh, Lowry et al. 2000; Lopez-Lago, Lee et al. 2000; Heo, Thapar et al. 2005). In the solution structure of the autoinhibited DH domain, the Ac region forms an amphipathic helix that binds in the GEF active site, blocking access to substrate (Aghazadeh, Lowry et al. 2000). The helix is melted and displaced, concomitant with activation of GEF activity, by phosphorylation of Tyr174, a residue in the Ac region that is completely buried in the DH-helix interface (Aghazadeh, Lowry et al. 2000). Although Tyr174 phosphorylation is slowed in an Ac-DH protein (AD) relative to an isolated Ac peptide (15-fold decrease in k_{cat}/K_M for the Src-family kinase Lck (Amarasinghe

and Rosen 2005)), reactivity of the key regulatory site remains appreciable despite its complete protection from solvent in the autoinhibited structure.

Here we examine the relationship between dynamics, regulation and activity in the AD module of murine Vav1. We demonstrate using NMR spectroscopy that the AD module exists in equilibrium between a ground state where the inhibitory helix is bound to the DH domain, and a weakly populated excited state where the helix is released and unfolded. Using a series of mutants with a wide range of equilibrium constants we show that the rate of Tyr174 phosphorylation by Lck is linearly related to the population of the excited state. Additionally, through GEF assays on representative proteins, we demonstrate that the catalytic activity of the AD protein is also dependent on the population of the excited state. In the full regulatory apparatus of Vav composed of the first five domains (CADPZ), interactions among the other domains are thermodynamically coupled to the core. The equilibrium is further biased ~10-fold toward the inhibited (helix bound) state in CADPZ and CADPZ_{K208A} versus in AD and AD_{K208A}, respectively. Concomitantly their relative GEF activities decrease by ~10-fold. Thus internal dynamics are required for and control both basal activity and the rate of full activation of the autoinhibited DH domain.

Results

The AD module undergoes motions on the μ s-ms timescale

Tyr174 is completely buried in helix-DH interface in the solution structure of the AD protein. However, the rate of Tyr174 phosphorylation by the Src-family kinase Lck is appreciable (Amarasinghe and Rosen 2005). These observations indicate that phosphorylation occurs through a distinct conformation in which the tyrosine sidechain is accessible to kinase. This conformation could be an excited state of the AD protein that is appreciably populated through a dynamic equilibrium, and simply captured by the kinase. Alternatively, binding of kinase to the helix-bound ground state could be necessary to alter the energy landscape of the system to favor a Tyr174-accessible conformation. In either scenario, the accessible state could involve large-scale conformational changes in the protein, such as dissociation of the Ac helix, or more local changes that expose the Tyr174 sidechain. To distinguish between these possibilities, we examined quantitative relationships between dynamics, phosphorylation kinetics and catalytic activity of the AD protein.

Conformational dynamics on μ s-ms timescales can be readily observed using NMR spectroscopy through measurement of relaxation dispersion, the variation of transverse relaxation rate, R_2 , with applied magnetic field (Palmer, Kroenke et al. 2001; Kay 2005). To assess dynamics in the AD protein, we measured single quantum ^{13}C relaxation dispersion of valine, leucine and

isoleucine ($\delta 1$ only) methyl groups (Lundstrom, Vallurupalli et al. 2007). Representative dispersion curves are shown in Figures 2-1a and Figure 2-2. Of 56 methyl resonances that could be analyzed, 25 show a difference (ΔR_2) between R_2 at the lowest and highest transverse magnetic fields of $> 2.0 \text{ s}^{-1}$ (Fig. 2-1c). These residues are located throughout the inhibitory helix, its binding site on the DH domain and in adjacent regions in the core of the DH domain (Fig. 2-1b), suggesting the arm is dynamic in the autoinhibited protein. Phosphorylation of AD on Tyr174 (to give pAD) quenched exchange ($\Delta R_2 < 2.0 \text{ s}^{-1}$ or $< 5.0 \%$ of non-phosphorylated value) of 11 methyl resonances, eight of which are located in the inhibitory arm or its interface with the DH domain (Figs. 2-1b, 2-1c and Fig. 2-3). These data on AD and pAD are consistent with a hypothesis that in the non-phosphorylated state the arm exists in a conformational equilibrium, and that this equilibrium is strongly biased upon phosphorylation. For the remaining 14 resonances, ΔR_2 remains large upon phosphorylation, and increases in 12 cases (Fig. 2-1c and Fig. 2-2). These 14 resonances are located in the core of the DH domain and on the backside of the protein opposite the active site (Fig. 2-3). The same resonances show nearly identical ΔR_2 in a DH protein lacking the Ac arm entirely ($\Delta A-D$, residues 181-375) as in pAD (Fig. 2-4). The different behaviors of these methyl resonances suggest that two dynamic processes with μs -ms timescales exist in the AD protein. One process involves fluctuations of the inhibitory arm; its effects on transverse relaxation are largely eliminated when the

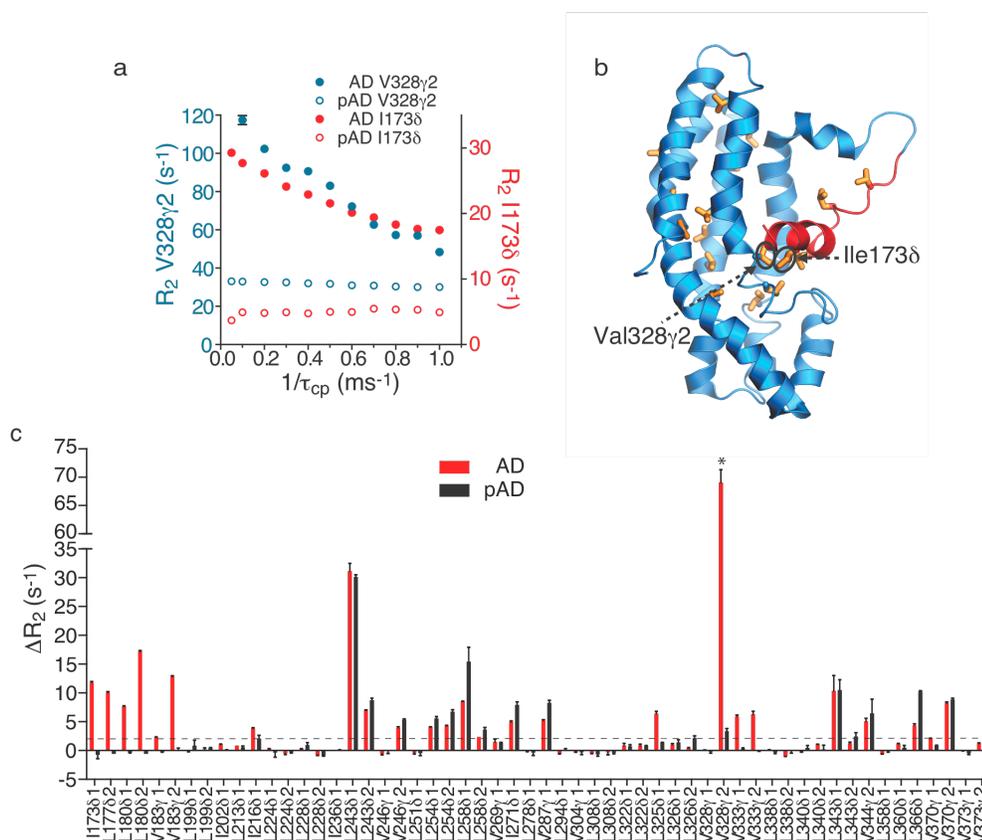


Figure 2-1. Relaxation dispersion in AD and structural mapping of dynamic residues. **(a)** Methyl single quantum ¹³C relaxation dispersion curves of Ile173 δ (red), Val328 γ 2 (blue) for AD (filled circle) and pAD (open circle). Error bars indicate s.d.; where they are not shown, standard deviation is smaller than the symbols. **(b)** Ribbon diagram of the AD structure (pdbID, 1F5X). Sticks show methyl containing sidechains with $\Delta R_2 > 2.0$ s⁻¹. Structure rendered using Pymol (<http://pymol.sourceforge.net/>). **(c)** Relaxation dispersion in AD and pAD. Bars show the differences (ΔR_2) in single quantum ¹³C R_2 between the lowest (50 Hz) and highest (1000 Hz) transverse fields of methyl groups in AD (red) and pAD (black). Due to the observation of multiple, thermodynamically coupled dynamic processes in AD (see text), it is not justified to fit CPMG relaxation dispersion data to a simple two-state model in order to obtain R_{ex} , the true relaxation dispersion amplitude. Therefore, ΔR_2 is employed throughout the chapter to assess relaxation dispersion amplitude of AD proteins.

*: Due to severe line broadening, ΔR_2 of Val328 γ 2 in AD is calculated from R_2 values at 100 Hz and 1000 Hz transverse fields collected under identical experimental conditions except that the constant relaxation period is 20 ms rather than 40 ms.

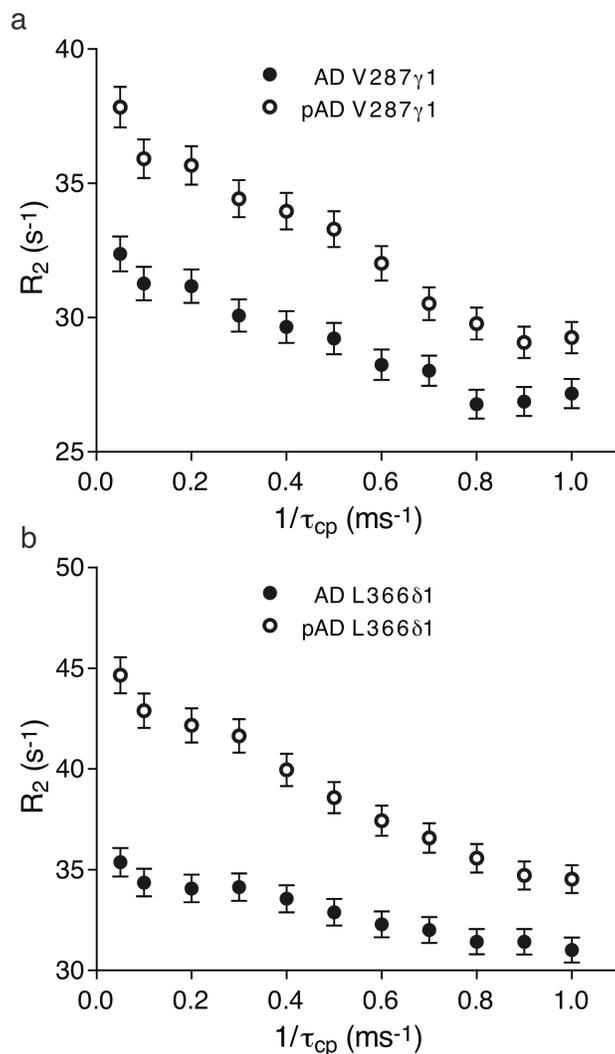


Figure 2-2. Single quantum ¹³C methyl relaxation dispersion curves for representative resonances that show increased relaxation dispersion amplitude upon phosphorylation. Data are shown for Val287 γ 1 (panel **a**) and Leu366 δ 1 (panel **b**) in AD (solid circle) and pAD (open circle). Error bars are obtained from the average of the standard deviations of at least two duplicate points if this average is above 2.0 % of the R_2 value, and are set to 2.0 % of the corresponding R_2 if below.

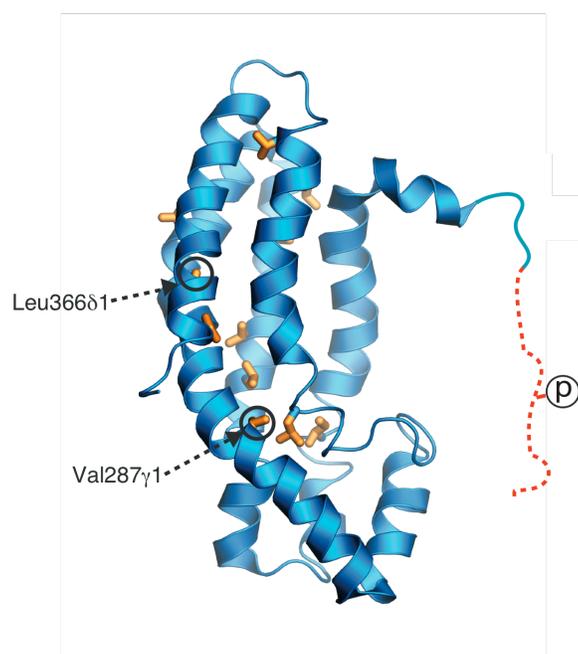


Figure 2-3. Ribbon diagram of the pAD structure model based on AD solution structure (pdbID, 1F5X1) with methyl-containing sidechains showing $\Delta R_2 > 2.0 \text{ s}^{-1}$ drawn as sticks. The sidechains of Val287 γ 1 and Leu366 δ 1, whose resonances are shown in Supplementary Figure 1 are circled. Structure rendered using Pymol².

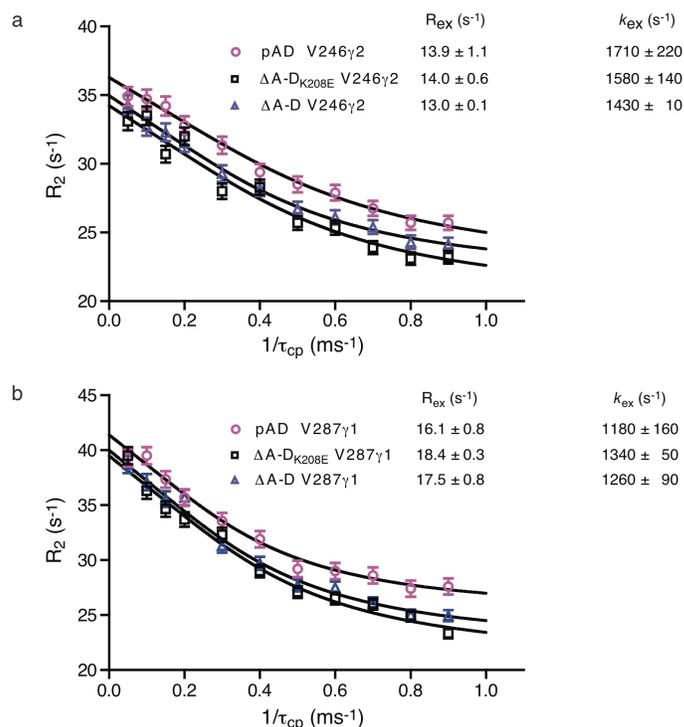


Figure 2-4. Single quantum ^{13}C methyl relaxation dispersion curves for pAD and the $\Delta A-D$ (residues 181-375, containing only the DH domain) and $\Delta A-D_{K208E}$. Data are shown for Val246 γ 2 (panel **a**) and Val287 γ 1 (panel **b**) in pAD (pink open circle), $\Delta A-D$ (blue open triangle) and $\Delta A-D_{K208E}$ (black open square). Both resonances have large exchange contributions from the dynamic process that is intrinsic to the DH domain (see main text and Supplementary Methods text for discussion of four-state dynamics in AD), and thus provide a sensitive comparison of the three proteins. Data were acquired at 10 °C on a 600 MHz instrument. Error bars are obtained from the average of the standard deviations of the four duplicate points if this average is above 2 % of the R_2 value, and are set to 2 % of the corresponding R_2 if below. Data were fit to a two-state model to yield values for R_{ex} , the exchange contribution to R_2 , and k_{ex} , the sum of the forward and reverse rate constants. Correspondence of the raw data and fitted values between pAD and $\Delta A-D$ indicates that the former is a good mimic of a protein lacking the inhibitory helix entirely, and thus of a fully open state. This conclusion is also supported by the fact that methyl ^1H and ^{13}C chemical shifts of $\Delta A-D$ are identical within experimental error to those from the DH domain of pAD, with the exception of peaks from Val183 γ 1, γ 2 (which probably are affected by the nearby charged N-terminus in the shorter construct). Correspondence of the raw data and fitted values between $\Delta A-D$ and $\Delta A-D_{K208E}$ indicates that the mutation does not affect the equilibrium intrinsic to the DH domain. Again, this conclusion is also supported by the similarity between $\Delta A-D$ and $\Delta A-D_{K208E}$ chemical shifts (not shown).

helix is displaced from the DH domain and melted by phosphorylation. The second process is intrinsic to the DH domain and persists even when the helix is dissociated. Its structural origins are currently unknown. Importantly, the change in relaxation dispersion for resonances in the second group upon phosphorylation indicates that the two dynamic processes are energetically coupled. That is, the thermodynamic and/or kinetic parameters of the second process are different depending on whether the helix is bound to the DH domain or not. Thus, quantitative analysis of the dispersion data requires modeling a minimum of four states (Fig. 2-5a). This assessment is consistent with our observation that fitting of the dispersion curves of individual resonances with a two-state model results in widely varying k_{ex} values (data not shown), a result that is a hallmark of multi-state (>2) equilibria (Korzhnev, Salvatella et al. 2004; Korzhnev, Neudecker et al. 2005).

The inhibitory arm fluctuates between bound and free states

In two-state equilibria, quantitative analysis of relaxation dispersion data can yield populations of states, rates of transitions between them and, in favorable circumstances, structural information in the form of chemical shifts of the weakly populated state (Palmer, Kroenke et al. 2001; Kay 2005). However, we (Li, Martins et al. 2008) and others (Korzhnev, Neudecker et al. 2005) have shown through simulations that when systems with three or more interconverting states are modeled assuming a two-state approximation, fitted parameters will be

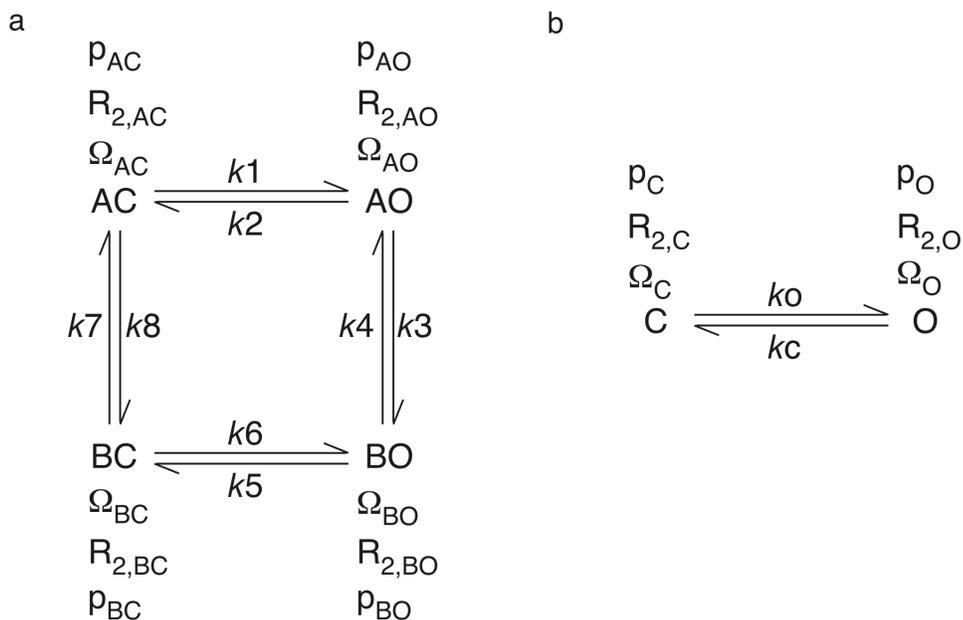


Figure 2-5. Models for the four-state equilibrium and the simplified two-state equilibrium involving the Ac helix in the AD protein. **(a)** The system is described by two coupled processes, one involving dissociation of the inhibitory helix, and a second within the DH domain itself. AC and BC represent the helix-bound (closed) states of the system; AO and BO represent the helix-dissociated (open) states. The dynamics within the DH domain are modeled as a second two-state equilibrium between states A and B. Fluctuations across this second process thus convert AO/AC to BO/BC. **(b)** In the simplified model, only the process involving dissociation of the inhibitory helix is depicted. In both **(a)** and **(b)**, $R_{2,i}$ is the transverse relaxation rate of species i in the absence of exchange. p_i is the fractional population of species i . Ω_i is the chemical shift of species i . The k s are first-order rate constants for the indicated transitions.

in error in the majority of situations. For this reason, extraction of thermodynamic and kinetic information from our CPMG data on AD, where a proper description requires a four-state analysis, is technically challenging and of uncertain feasibility. We chose to tackle this challenge separately (See Chapters 3 and 4 for details).

Thus, to characterize the structural and energetic features of the motions in AD, we adopted an alternative strategy based on the fact that a protein in fast exchange between two states will have NMR chemical shifts that are population weighted averages of the chemical shifts of the pure states. Thus, a series of mutants that differentially sample an equilibrium will have resonances that lie along a straight line between the pure states. Although originally developed for qualitative analysis of a simple two-state system (Volkman, Lipson et al. 2001), this approach is also effective in quantitatively characterizing more complex systems like the four-state equilibrium in AD as long as the process of interest can be selectively perturbed (Derivation 1). We created a series of AD mutants designed to decrease helix-DH affinity: AD_{K208E}, AD_{K208S}, AD_{K208A}, AD_{E169K}, AD_{K208A}, AD_{K208V} and AD_{E169K K208V} (see Methods). We designed a second set to increase helix-DH affinity through increased electrostatic interactions between the helix and DH domain (Selzer, Albeck et al. 2000): AD_{E169K} and AD_{E169K E207K}. In ¹H/¹³C HSQC spectra, most methyl resonances are similar in the wild type and mutant proteins, suggesting that the mutations did not cause major structural

changes to the DH domain. All resonances that show large chemical shift differences also show large ΔR_2 . Importantly, the chemical shifts of these dynamic resonances lie on a straight line between the corresponding peaks in $AD_{E169K E207K}$ (designed to have the most stable helix-DH interactions (Selzer, Albeck et al. 2000)) and Tyr174-phosphorylated AD (pAD; Figs. 2-6 and 2-7). Moreover, for a given protein, the fractional deviation from $AD_{E169K E207K}$ and pAD is very similar for all shifting peaks (Fig. 2-6 and 2-7). This observed colinearity is a strong indicator that the proteins differentially sample a common equilibrium between a structural state resembling $AD_{E169K E207K}$ with the inhibitory helix bound to the DH domain, and another state resembling pAD with the helix dissociated and unfolded.

Quantitative analysis of dynamics in the AD protein

Two pieces of data indicate that pAD closely approximates an idealized open state, where the population of the helix bound to the DH domain is immeasurably small. First, methyl 1H and ^{13}C chemical shifts of $\Delta A-D$ are identical within experimental error to those from the DH domain of pAD (chemical shift differences in ^{13}C dimension are smaller than half the digital resolution). The only exceptions to this agreement are peaks from Val183 γ 1, γ 2, which likely are affected by the nearby charged N-terminus in the shorter construct. Second, the relaxation dispersion behavior of DH domain resonances

Derivation 1: Linear behavior of chemical shift in perturbations of a four-state equilibrium

Statement: the resonances derived from fast four-state equilibrium will be collinear given the assumptions that mutations only directly perturb kinetics and thermodynamics of the two edges involving opening-closing (AC-AO and BC-BO) and not the other two edges involving the second dynamic process (AC-BC and AO-BO) and mutations do not change chemical shift of any of the states.

Evidence supporting these assumptions:

First, the sites of our mutations are directly under the helix at its interface with the DH domain. They are far from the resonances that are strongly affected by the DH dynamics (i.e. that show large ΔR_2 in the phosphorylated protein). Thus, from a structural standpoint the mutations should affect the helix dynamics much more than the DH dynamics. Second, $\Delta A-D$ and $\Delta A-D_{K208E}$ have identical chemical shifts (data not shown) and relaxation behavior (compare black open squares ($\Delta A-D_{K208E}$) and blue open triangles ($\Delta A-D$) in Fig. 2-4). If the K208E mutation were directly affecting the DH dynamics and structures of the different states, these NMR properties would be different between mutant and wild type proteins. Thus, even in the mutant that has the strongest effect on the dynamics of the AD protein, we can ascribe this effect with great confidence to selective perturbation of the helix equilibrium.

Refer to Figure 2-5 for definitions of all parameters used below.

The observed ^{13}C resonance derived from a methyl group in a particular construct i is given by:

$$\Omega_{\text{observed}}^i = p_O^i * \Omega_O^i + p_C^i * \Omega_C^i \quad (\text{A1})$$

where p_O^i , p_C^i are the open and closed fractional population of the construct i and the sum of p_O^i and p_C^i is unity; Ω_O^i and Ω_C^i are the effective open and closed state chemical shifts given by:

$$\Omega_O^i = [p_{AO}^i / (p_{AO}^i + p_{BO}^i)] * \Omega_{AO}^i + [p_{BO}^i / (p_{AO}^i + p_{BO}^i)] * \Omega_{BO}^i \quad (\text{A2})$$

$$\Omega_C^i = [p_{AC}^i / (p_{AC}^i + p_{BC}^i)] * \Omega_{AC}^i + [p_{BC}^i / (p_{AC}^i + p_{BC}^i)] * \Omega_{BC}^i \quad (\text{A3})$$

where index i represents that the parameters are specific for the construct i .

According to the assumptions in the statement, none of the parameters on the right hand side of (A2) and (A3) are affected by mutations and therefore the resulting left hand sides of (A2) and (A3) are mutation independent and the index i of Ω_O^i and Ω_C^i in (A1), (A2), (A3) can be removed. Therefore (A1) can be rewritten as:

$$\Omega_{\text{observed}}^i = p_O^i * \Omega_O + p_C^i * \Omega_C \quad (\text{A4})$$

which places $\Omega_{\text{observed}}^i$ on the straight line defined by the two extremes, Ω_O and Ω_C .

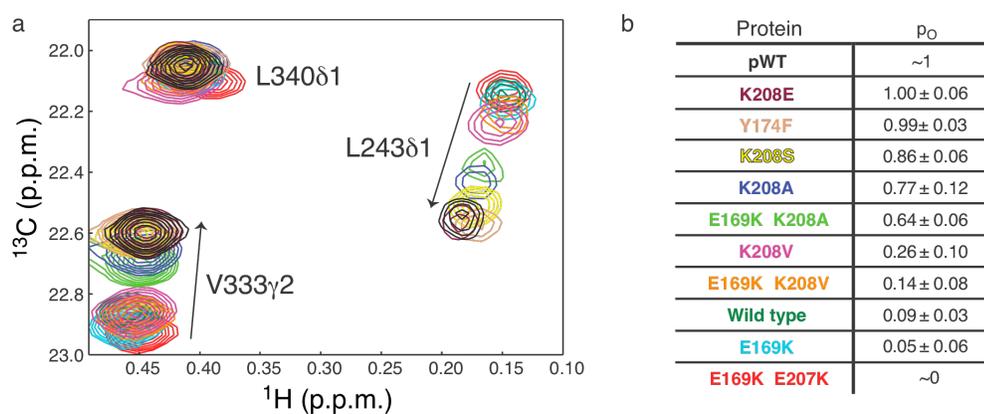


Figure 2-6. AD mutants differentially sample the helix-DH equilibrium. (a) Representative peaks in superimpositions of $^1\text{H}/^{13}\text{C}$ HSQC spectra of AD mutants. Arrows point in the direction of increasing population of the open state. Peaks are colored: pAD (black), AD_{K208E} (burgundy), AD_{Y174F} (light brown), AD_{K208S} (yellow), AD_{K208A} (blue), $\text{AD}_{\text{E169K K208A}}$ (light green), AD_{K208V} (pink), $\text{AD}_{\text{E169K K208V}}$ (orange), AD wild type (dark green), AD_{E169K} (cyan), and $\text{AD}_{\text{E169K E207K}}$ (red). (b) Open populations (p_o , average \pm standard deviation) of AD proteins.

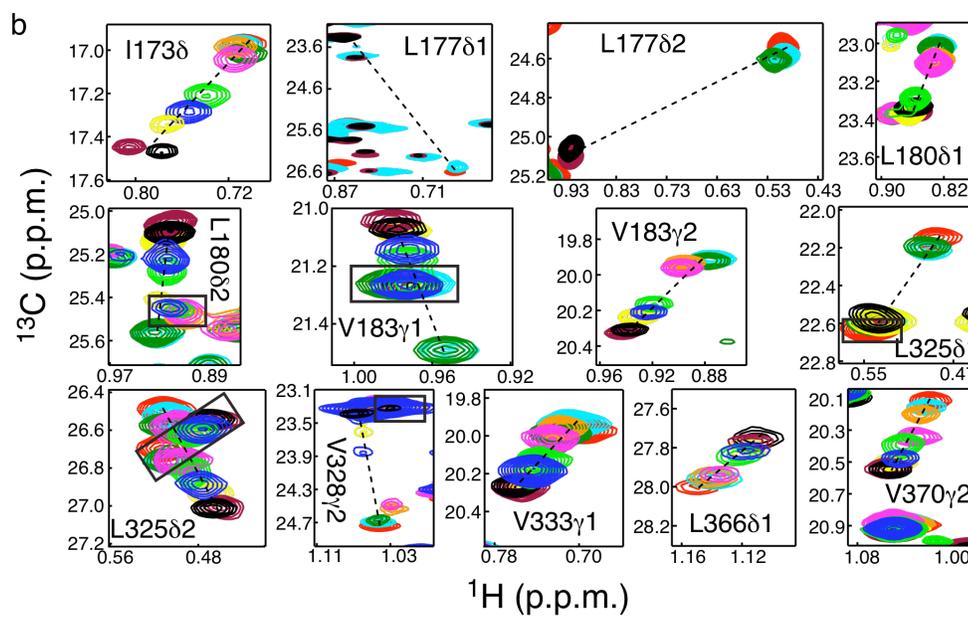
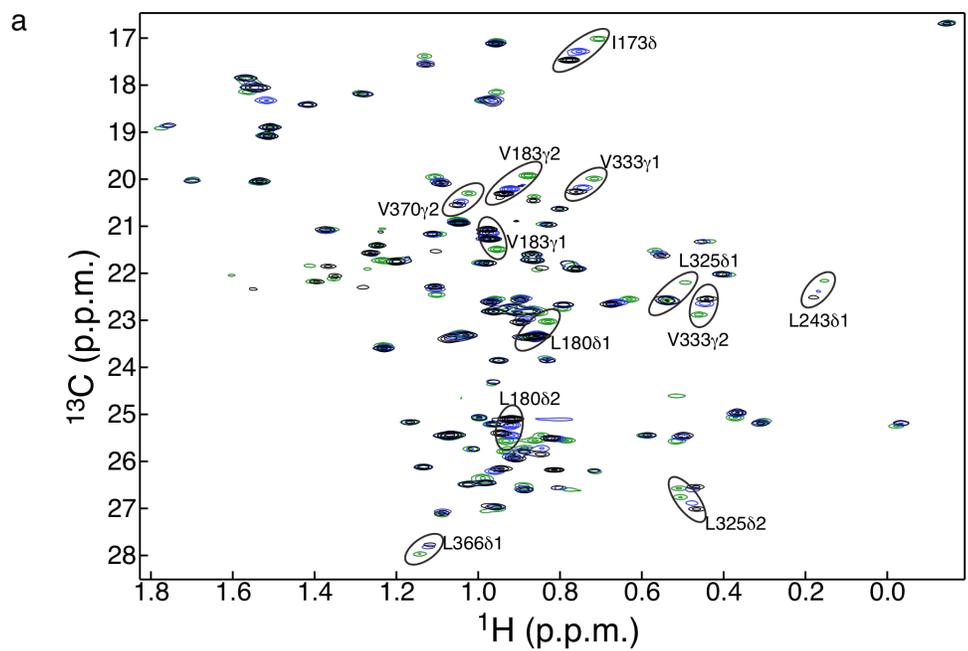


Figure 2-7. Overlay of $^1\text{H}/^{13}\text{C}$ HSQC spectra. (a) Methyl region of $^1\text{H}/^{13}\text{C}$ HSQC spectra (Ala β , Val γ 1/2, Leu δ 1/2 and Ile γ 2) of AD (dark green), AD_{K208A} (blue), and pAD (black) are shown. Note that the fractional displacement of resonances in AD_{K208A} relative to AD and pAD is very consistent among the various peaks. (b) Expansions of overlays for all peaks used in the analysis of Figure 2-9, except for those shown in Figure 2-6. Dotted lines along peak shift trajectory help identify the relevant peaks in the selected regions. Peaks are colored according to Figure 2-6. Boxes indicate overlapping resonances that are not part of the mutant trajectory.

in the ΔA -D and pAD proteins is virtually identical, except for a small, uniform increase $R_{2,0}$ (the exchange-independent value of R_2) for all resonances in pAD due to the slower overall tumbling time of this larger protein (Fig. 2-4). Coincidence is observed both for resonances that show large ΔR_2 in pAD and those that do not. Since relaxation dispersion is highly sensitive, even to small populations ($\sim 1\%$) of alternative states, we conclude that pAD is a reasonable mimic of an idealized open state where the helix is fully dissociated from the DH domain.

Related data indicate that the mutant $AD_{E169K E207K}$ closely approximates the opposite idealized state, in which the helix is fully bound to the DH domain. This mutant was designed to maximally increase affinity of the inhibitory helix for the DH domain. It has methyl chemical shifts similar to those of wild type AD, but has dramatically decreased ΔR_2 for residues in the inhibitory helix and its DH binding site (average ΔR_2 of $2.1 \pm 1.9\text{ s}^{-1}$ relative to $8.1 \pm 1.9\text{ s}^{-1}$ in WT based on the six methyl resonances in the inhibitory helix and the adjacent loop region, Ile173 δ , Leu177 δ 2, Leu180 δ 1, Leu180 δ 2, Val183 γ 1 and Val183 γ 2). In highly biased equilibria, relaxation dispersion is much more sensitive than chemical shift to changes in equilibrium populations (Derivation 2). Thus, these data indicate that the closed conformation is heavily populated in both AD and $AD_{E169K E207K}$, but that the latter protein has a more strongly skewed equilibrium. The very small magnitude ΔR_2 observed for $AD_{E169K E207K}$ typically indicates an excited state

population $< \sim 1-2\%$. Thus, we conclude that the chemical shifts of AD_{E169K E207K} approximate those of an idealized closed state of the system.

With pAD and AD_{E169K E207K} as endpoints in the open-closed equilibrium, we then used the methyl chemical shifts of each mutant to calculate its populations of the open and closed states, p_o and p_c respectively. As shown in Figure 2-6, p_o varied from 0.05 to 1.00 across the series.

Phosphorylation occurs through the excited state

To understand how the equilibrium in the autoinhibited DH domain affects its regulation, we first compared the second order rate constant (k_{cat}/K_M) for Lck-mediated phosphorylation of an Ac region peptide (residues 169-190) with that of the most open mutant, AD_{K208E} ($p_o = 1.00$). The two substrates possess very similar kinetic parameters ($k_{cat}/K_M = 9.7 \pm 0.7 \times 10^5 \text{ min}^{-1} \text{ M}^{-1}$ (peptide) vs. $9.0 \pm 0.4 \times 10^5 \text{ min}^{-1} \text{ M}^{-1}$ (AD_{K208E})), indicating that the DH domain does not substantially affect the phosphorylation kinetics in the open conformation. We then performed analogous measurements of all other mutants (Figs. 2-8 and 2-9). For each construct, phosphorylation rate scales linearly with concentration of kinase (not shown), demonstrating that the opening and closing kinetics of the inhibitory helix are not rate limiting in the reaction. We found that k_{cat}/K_M varies over a large range for the different proteins and is linear with p_o determined by NMR (Fig. 2-9). Thus, as predicted from the structure of AD, the helix-bound state is phosphorylated very poorly, if at all (intercept ~ 0 in Fig. 2-9).

Derivation 2: In highly biased equilibria relaxation dispersion is much more sensitive than chemical shift to changes in equilibrium populations.

For simplicity, we take two-state equilibrium ($A \leftrightarrow B$) as an example to demonstrate this statement. In fast exchange:

$$\Omega_{\text{observed}} = p_A * \Omega_A + (1-p_A) * \Omega_B = p_A * (\Omega_A - \Omega_B) + \Omega_B$$

So,

$$d(\Omega_{\text{observed}})/d(p_A) = (\Omega_A - \Omega_B)$$

which is constant for all values of p_A .

The exchange contribution to R_2 , R_{ex} , is given by:

$$R_{\text{ex}} = (p_A * p_B)(\Omega_A - \Omega_B)^2/k_{\text{ex}} = (p_A)(1-p_A)(\Omega_A - \Omega_B)^2/k_{\text{ex}}$$

So,

$$d(R_{\text{ex}})/d(p_A) = (1-2p_A)(\Omega_A - \Omega_B)^2/k_{\text{ex}}$$

which is dependent on p_A , and has largest absolute value as p_A approaches 0 or 1.

Thus, R_{ex} changes most with p_A for small or large values of p_A , while changes in chemical shift with p_A are constant across the entire $\Delta\omega$ range. For example, when $p_A = 0.96$, a population change of 0.02, to $p_A = 0.98$, will introduce a 2% change in chemical shift but a 50% change in R_{ex} .

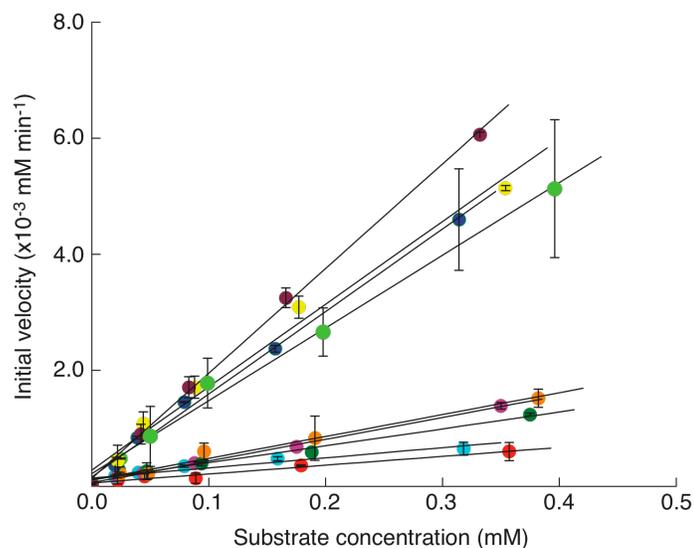


Figure 2-8. Vav mutants show different phosphorylation rates. Initial phosphorylation rate versus substrate concentration is shown for a series of AD constructs. Phosphorylation data were collected in the linear region of the Michaelis-Menten curve at substrate concentrations below K_M for the most open construct (AD_{K208E}). Data for the following DH domain mutants are shown: K208E (burgundy), K208S (yellow), K208A (blue), E169K K208A (light green), K208V (pink), E169K K208V (orange), wild type (dark green), E169K (cyan), and E169K E207K (red).

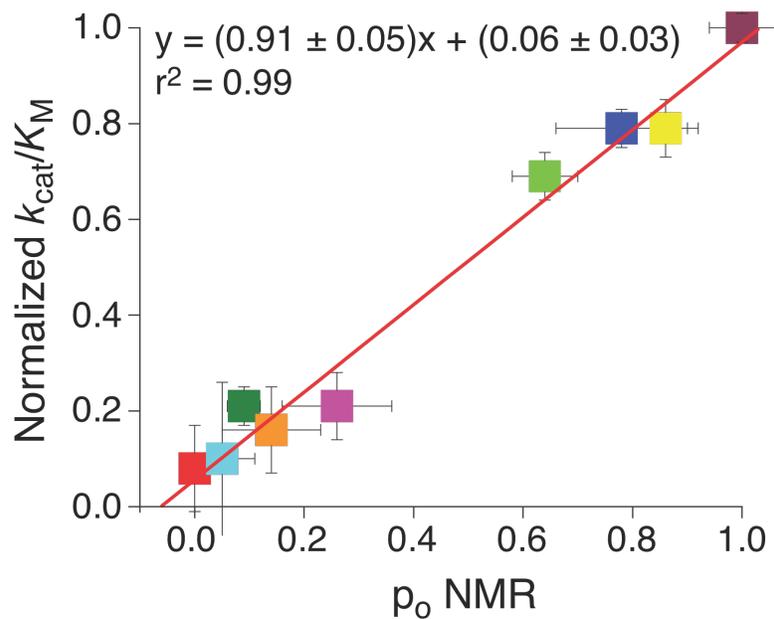


Figure 2-9. The conformational equilibrium of the inhibitory helix controls the rate of AD phosphorylation. For each mutant, the normalized rate constant for phosphorylation $\{(k_{\text{cat}}/K_M)_{\text{mutant}}/(k_{\text{cat}}/K_M)_{\text{K208E}}\}$ of Tyr174 by Lck is plotted against p_o , determined by NMR (Fig. 2-6). Rate constants are the average of at least two independent measurements; error bars represent standard error. The error bars for p_o show the standard deviation. Line shows the best linear least squares fit of the data. Data colored as in Figure 2-6a.

Phosphorylation occurs through transient excursions to an excited state where the inhibitory helix is melted and Tyr174 is accessible to kinases. Since the conformational transitions of the helix are rapid, the overall phosphorylation rate is directly proportional to the fractional population of the open state (slope ~ 1 in Fig. 2-9).

Helix-DH conformational equilibrium controls GEF activity

To understand how the helix-DH dynamics affect catalytic function, we examined the GEF activities of WT and mutant AD proteins. In order to separate effects of the mutations on the helix-DH equilibrium from effects on intrinsic DH activity, in each case we compared the corresponding AD and ΔA -D proteins. Our initial goal was to analyze all of the mutants in Figure 2-6. However, we found that most amino acid changes at position 208, other than K208A, severely diminish the intrinsic GEF activity of the DH domain (i.e. in the ΔA -D proteins). Therefore our analysis here is limited to AD, AD_{Y174F} and AD_{K208A}. As shown in Figure 2-10, the AD, AD_{Y174F} and AD_{K208A} proteins have GEF activities that are 7 ± 2 %, 81 ± 19 % and 80 ± 17 % of the corresponding ΔA -D proteins, respectively. These values are consistent with their respective open populations of 9 ± 3 %, 99 ± 3 % and 77 ± 12 % measured by NMR (Fig. 2-6b). Thus, the helix-DH equilibrium also controls the basal GEF activity of the autoinhibited AD proteins.

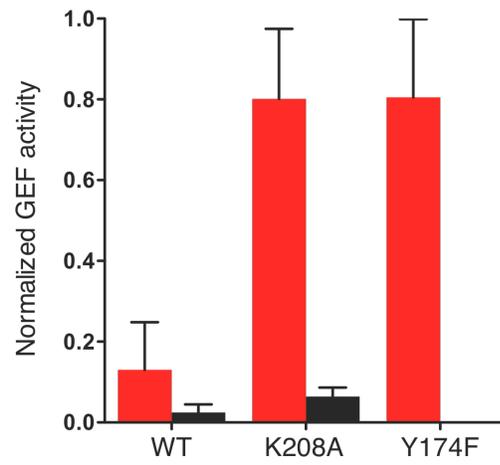


Figure 2-10. Conformational equilibrium of the inhibitory helix controls GEF activity in AD (red) and CADPZ (black). See Methods for measurement and normalization procedures.

CH interactions cooperatively suppress DH GEF activity in CADPZ

In full length Vav, the AD element is flanked N-terminally by a Calponin homology (CH) domain and C-terminally by pleckstrin homology (PH) and zinc finger (ZF) domains. Interactions among these additional domains also contribute to suppression of GEF activity, and together with AD form a cooperative element (CADPZ) that controls the catalytic activity of Vav (Bustelo 2001). To understand how the domains outside the core contribute to autoinhibition, we compared AD to CADPZ. In methyl TROSY spectra, the resonances representing the inhibitory helix and DH domain are in very similar positions in both constructs (not shown). Thus, the basic structure of the AD core is preserved in the presence of the additional domains in the larger protein. We focused our analysis on the ten methyl groups that are either in the Ac helix or contact the helix directly at the inhibitory interface. Of these resonances, seven are sufficiently free of overlap in CADPZ to be analyzed quantitatively. As shown in Figure 2-11a, multiple quantum $^1\text{H}/^{13}\text{C}$ ΔR_2 of all these methyl groups is significantly reduced in CADPZ relative to AD. We also examined methyl TROSY spectra of fully phosphorylated CADPZ (pCADPZ, see Methods), a Y174D mutant of CADPZ (CADPZ_{Y174D}) that mimics the phosphorylated protein, and one CADPZ mutant corresponding to K208A produced for AD (Fig. 2-11b). Methyl chemical shifts in pAD, pCADPZ and CADPZ_{Y174D} are nearly identical (data not shown), indicating that phosphorylation also shifts the helix-DH

equilibrium in CADPZ to strongly favor the dissociated state. In the mutant, five of the interface methyl resonances are in fast exchange in CADPZ. As in AD, these resonances lie along the linear trajectory between the non-phosphorylated and phosphorylated chemical shifts (Fig. 2-11b). Thus, at least from the standpoint of the inhibitory interface, CADPZ exists in a regulatory equilibrium analogous to that in AD, fluctuating between helix-bound and helix-dissociated states. For CADPZ_{K208A}, the equilibrium is shifted in CADPZ toward the helix-bound state relative to AD (Fig. 2-11b), consistent with the reduction of ΔR_2 in CADPZ observed above. On average the equilibrium is shifted ~10-fold, corresponding to a free energy change of $\sim 1.4 \text{ kcal mol}^{-1}$ (Fig. 2-11c). The shift in equilibrium also is manifest in GEF activity. As in the AD proteins, GEF activity of CADPZ is appreciably lower than that of CADPZ_{K208A}, reflecting destabilization of autoinhibitory contacts in the mutant (black bars in Fig. 2-10). Most importantly, the activities of the CADPZ proteins are significantly below those of their AD counterparts, reflecting suppression of the helix-DH equilibrium in the five-domain system (compare black and red bars in Fig. 2-10). Thus, contacts of domains outside the AD core (likely involving the CH domain) modulate the equilibrium in the core and thereby control GEF activity.

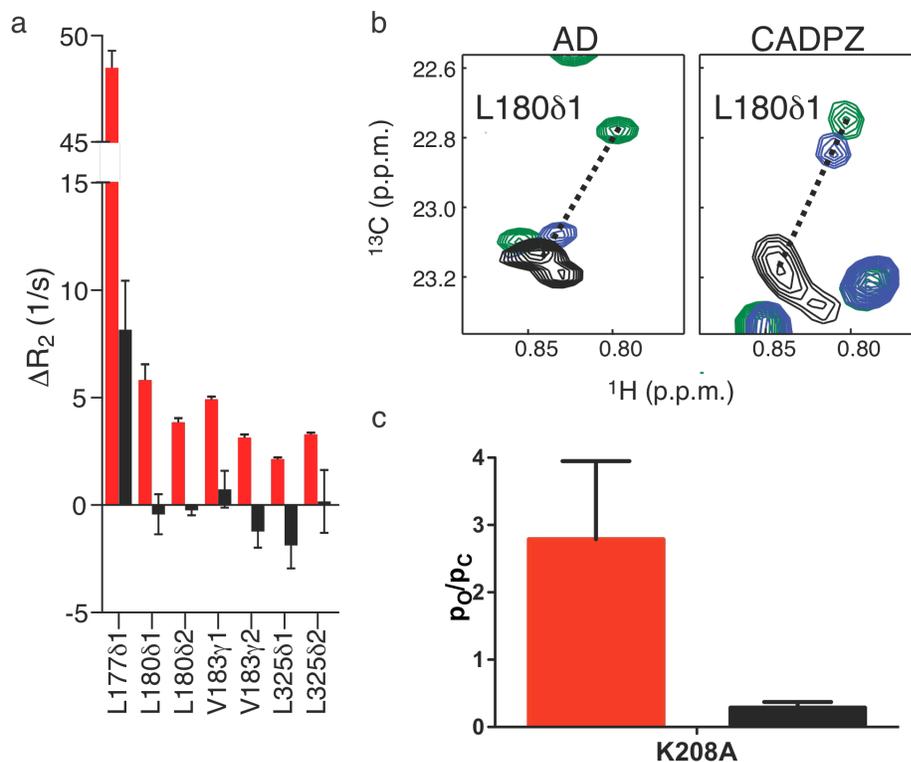


Figure 2-11. Interdomain cooperativity in Vav1. **(a)** Multiple quantum $^1\text{H}/^{13}\text{C}$ ΔR_2 for residues in the inhibitory interface for AD (red bars) and CADPZ (black bars). Error bars indicate standard deviation for two independent measurements. **(b)** Overlaid $^1\text{H}/^{13}\text{C}$ methyl TROSY spectra for WT (green) and K208A (blue) mutants and pAD/CADPZ_{Y174D} (black) proteins for AD (left panels) and CADPZ (right panels), showing Leu180 δ 1 (top panels). Dotted lines connect resonances from phosphorylated and non-phosphorylated proteins. **(c)** p_O/p_C for AD_{K208A} (red bars) and CADPZ_{K208A} (black bars) mutants. Error bars indicate the propagated standard deviation for p_O/p_C .

Discussion

Together the NMR and biochemical data support the model for function and regulation of the AD element of Vav1 shown in Figure 2-12. The AD protein exists in an equilibrium between a ground state conformation where the inhibitory helix is bound to the DH domain as observed in the solution structure, and an excited state resembling pAD, where the helix is dissociated from the DH domain and melted. In the ground state, Tyr174 is buried in the interface of the inhibitory helix with the active site of the DH domain. These contacts block access of kinases to Tyr174 and of GTPase substrates to the DH active site. In the excited state, Tyr174 and the DH active site are accessible, and phosphorylation rate and GEF catalysis are those of the free Tyr174 site and free DH domain, respectively. A second equilibrium within the DH domain core (not shown in Fig. 2-12) is coupled to the helix dynamics; work is underway to determine the functional significance, if any, of this additional process. Under the conditions used in the assays in this chapter, the conformational transitions of the helix are rapid relative to phosphorylation and GEF catalysis, so for any given protein the measured rates of these activities are proportional to the population of the open state. In wild type AD, the equilibrium is strongly biased toward the helix-bound state (~90 %, Fig. 2-6b), and phosphorylation rate and GEF activity are low. Upon phosphorylation, the protein is driven strongly toward the helix-dissociated/melted state where it is fully active. Thus, internal dynamics of the

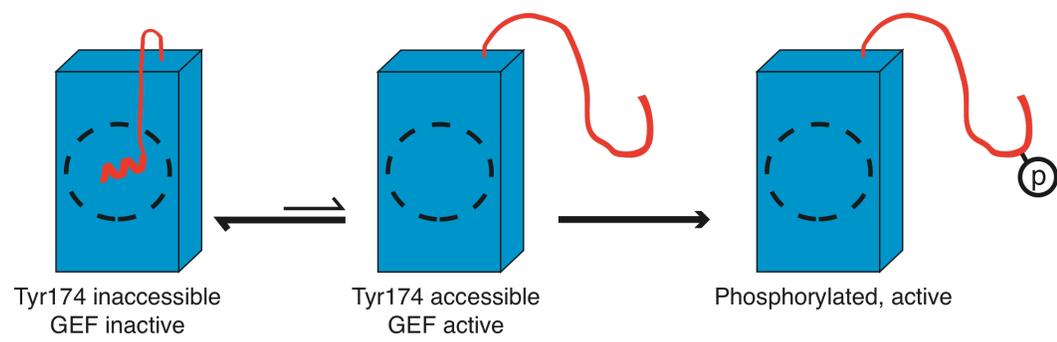


Figure 2-12. Model for function and regulation of the AD module of Vav1. See text for explanation.

autoinhibited AD module play a central role in dictating its basal catalytic activity and in enabling its full activation by tyrosine kinases.

Characterization of the open population of the autoinhibited AD protein provides an explanation for a puzzling observation in the literature: the Y174F mutant of full length Vav1 has appreciable transforming activity *in vivo*, even though it cannot undergo the activating phosphorylation event (and thus should be constitutively inactive) (Bustelo 2001). Our NMR analysis revealed that AD_{Y174F} is 99 % open, versus 9 % for the wild type protein (Fig. 2-6b). This change corresponds to c.a. 4.2 kcal mol⁻¹ decrease in the free energy of the open/active state relative to the closed/inactive state. Concomitantly, the Y174F mutation increases the relative GEF activity by about 11-fold in AD (Fig. 2-10), and a similar effect was seen with the full-length regulatory element (data not shown). Thus our data explain the enhanced activation of Rac and cell transformation induced by this mutation.

A recent EM reconstruction of Vav3 has suggested that the CH domain interacts with the ZF domain and perhaps the Ac region (Llorca, Arias-Palomo et al. 2005). These interactions appear to act on the AD element to cooperatively suppress the DH domain, since mutations to Vav that incapacitate or eliminate the CH domain lead to increased GEF activity and resultant cell transformation (Katzav, Martin-Zanca et al. 1989; Katzav, Cleveland et al. 1991; Zugaza, Lopez-Lago et al. 2002; Llorca, Arias-Palomo et al. 2005). Our characterization of the

AD element and the CADPZ element explains the effects of CH perturbations in quantitative terms. Interactions involving CH domain in CADPZ biases the core equilibrium ~10-fold towards the helix bound state via thermodynamic coupling (Fig. 2-13). In mutants where CH is truncated, even though the core autoinhibitory module (AD) is intact, the inherent thermodynamics of the module produce substantial populations of the active state and probably higher steady state phosphorylation level as well. When such truncated proteins are over-expressed, this population could easily be sufficient to cause transformation. This construction of Vav, with a core active site repression mechanism that is modulated by contacts of other modular elements, is often observed in autoinhibited multi-domain systems (Lim 2002; Pawson 2004). It is clear in many cases that the modulatory interactions both suppress activity in the basal state, and provide mechanisms of integrating multiple inputs to achieve signaling specificity *in vivo* (Lim 2002; Dueber, Yeh et al. 2003). However the energetic bases of this modulation have generally not been explored in the literature prior to this study, and could involve changes to the kinetics, thermodynamics and/or structure of the core and its interconverting states. Our characterization of the AD core of Vav1 sets the stage to understand how the physical properties of this element are modulated by interactions of other domains in the protein. Further characterization of CADPZ demonstrates the thermodynamic basis of this regulatory modulation in Vav, which provides the level of suppression and control

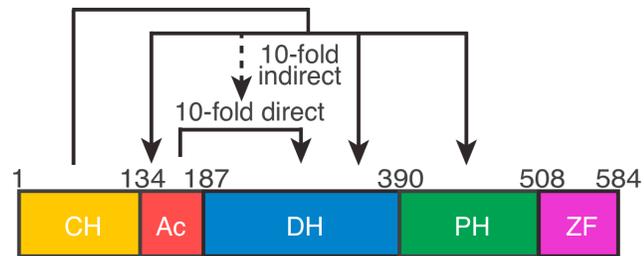


Figure 2-13. Domain architecture of CADPZ and thermodynamic model for its cooperative suppression. Domains are colored: CH (orange), Ac (red), DH (blue), PH (green), ZF (magenta). Solid lines represent direct contacts whereas dashed lines represent indirect interaction.

of DH activity necessary for *in vivo* function. Work to understand the structural basis of this regulatory modulation is currently underway in our lab. Clearly more similar work in other systems is needed to establish the generality of our observations.

Methods

Molecular biology and protein purification. We generated Vav mutants using the Quick Change Mutagenesis Kit (Stratagene), and verified them by DNA sequencing. We expressed all AD and Δ A-D proteins (residues 169-375 and 181-375, respectively) from pET11a (Novagen). We generated U-[^{15}N], Ile-[$^{13}\text{C}^{\gamma 2}\text{H}_3$]-, Leu-[$^{13}\text{CH}_3$, $^{13}\text{CH}_3$], Val-[$^{13}\text{CH}_3$, $^{13}\text{CH}_3$]-labeled proteins through bacterial growth in M9 media containing 1 g per liter of 3- ^{13}C pyruvate (added 1 hr before induction) (Korzhev, Kloiber et al. 2004), and U-[^{15}N , ^2H], Ile-[$^{13}\text{C}^{\delta}\text{H}_3$]-, Leu-[$^{13}\text{CH}_3$, $^{12}\text{CD}_3$], Val-[$^{13}\text{CH}_3$, $^{12}\text{CD}_3$]-labeled proteins through bacterial growth in M9 D₂O media containing 50 mg per liter 2-keto-3-methyl-d₃-3-d₁-4- ^{13}C -butyrate and 50 mg per liter 4-[^{13}C , ^1H]-3,3- ^2H - α -ketobutyrate (added 1 hr before induction) (Korzhev, Kloiber et al. 2004). Natural abundance proteins for biochemical studies were produced by bacterial growth in LB medium.

The Ac peptide (residues 169-190) and pAD were obtained as described previously (Amarasinghe and Rosen 2005). We purified all AD proteins except AD_{E169K E207K} using ion exchange chromatography and size exclusion chromatography as described by Amarasinghe and Rosen (Amarasinghe and

Rosen 2005). We purified the mutant AD_{E169K E207K} and ΔA-D using a phenyl sepharose FF column (GE Healthcare) in 1M (NH₄)₂SO₄, followed by ion exchange chromatography and size exclusion chromatography.

We co-expressed the Vav1 CADPZ proteins (residues 1-584) as maltose binding protein (MBP) fusions in *E. coli* strain BL21(DE3) with the groES-groEL chaperones (Takara Bio Inc). Chaperone expression was induced at OD₆₀₀ ~ 0.25 with 200 mg per liter of arabinose. Expression of the MBP-fusion was induced at OD₆₀₀ ~ 0.8 with 0.2 mM IPTG. We purified the CADPZ proteins by successive weak anion exchange, amylose affinity and strong anion exchange chromatographies, Tev protease cleavage to remove MBP, and successive strong anion exchange and gel filtration chromatographies.

Selection of residues for mutagenesis; Hypare analysis. Vav mutants were selected based on analysis of the AD solution structure (pdbID: 1F5X) using the Hypare software package (Selzer, Albeck et al. 2000). This software uses calculated electrostatic interactions to estimate the effect of mutations on the association rate constant between two binding partners of known structure (<http://bioportal.weizmann.ac.il/hypareb/main>). In order to obtain input for Hypare, we divided the inhibitory helix (residues 169-180) from the DH domain (181-375) of a single model from the AD solution structure ensemble (Aghazadeh, Lowry et al., 2000). Hypare analysis predicted that residues 169, 207 and 208 should have a strong effect on affinity of the Tyr174-containing peptide for the

DH domain. We initially screened a total of > 20 single and double mutants by $^1\text{H}/^{13}\text{C}$ HSQC spectra to assess the effect of the mutations on the overall structure of the DH domain and the inhibitory arm. Mutants displaying global changes judged by large non-uniform chemical shift changes in HSQC spectra were eliminated and the remaining mutants were analyzed biochemically and by NMR.

NMR spectroscopy. NMR samples contained 1.0 mM or lower freshly purified protein in 25 mM phosphate buffer (pH 7.2) with 50 mM NaCl, 5 mM DTT, 1 mM EDTA, 0.1 % (w/v) NaN_3 . All NMR experiments were recorded on 600 and 800 MHz Varian Inova spectrometers. The methyl ^{13}C CPMG relaxation dispersion pulse sequence was kindly provided by Dr. Lewis E. Kay (Tollinger, Skrynnikov et al. 2001; Mulder, Hon et al. 2002; Lundstrom, Vallurupalli et al. 2007). All CPMG relaxation dispersion experiments used a constant relaxation period of 40 ms except those collected at 800 MHz and 10 °C, which used a period of 20 ms. For all CPMG spectra, 1530 and 192 complex data points were acquired in the ^1H and ^{13}C dimensions, respectively. For all HSQC spectra, 1530 and 512 complex data points were acquired in the ^1H and ^{13}C dimensions, respectively. Spectra were acquired with $^1\text{H}/^{13}\text{C}$ sweep widths of 7000/2200 Hz and 9000/2500 Hz on the 600 MHz and 800 MHz instruments, respectively.

We calculated the average open populations (p_o) of the AD mutants from chemical shifts measured at 600 MHz and 15 °C. Resonances showing chemical shift differences between AD and pAD that are > 10x digital resolution (> 10 x

0.029 ppm) were analyzed further. We calculated open state population values according to equation 1:

$$p_o = \frac{\Omega_x - \Omega_c}{\Omega_o - \Omega_c} \quad (1)$$

where Ω_x , Ω_c , and Ω_o are the chemical shift of a given resonance in mutant X, AD_{E169K E207K} and pAD, respectively. Mean p_o values were determined from at least 10 resonances. We eliminated outlier p_o values (on average 1 per mutant) based on the Grubbs test, performed under a Dataplot session at 95 % confidence level, assuming a Gaussian distribution (Heckert 2003).

We calculated the average ΔR_2 for AD_{E169K E207K} and AD from the averaged R_2 difference between the lowest and the highest effective B_1 fields for 6 dynamic resonances in the inhibitory helix (Ile173 δ , Leu177 δ 2, Leu180 δ 1, Leu180 δ 2, Val183 γ 1 and Val183 γ 2).

Kinase assay. We measured tyrosine 174 phosphorylation kinetics using a spectrophotometric assay that couples the production of ADP to the oxidation of NADH stoichiometrically (Amarasinghe and Rosen 2005). Reactions were carried out on a Beckmann DU800 spectrophotometer equipped with a Peltier temperature control unit. Samples were preincubated at 15 °C and all reactions were carried out at 15 °C as described previously (Amarasinghe and Rosen 2005). The enzyme reactions contained 50 mM HEPES, pH 7.5, 50 mM NaCl, 10 mM

DTT, 2 mM ATP, 10 mM MgCl₂, 1 mM phosphoenolpyruvate, 0.2 mM NADH, 89 units per mL pyruvate kinase, 124 units per mL lactate dehydrogenase, and varying concentrations of substrate. All assays contained 20 nM Lck kinase. The Lck kinase construct contained a mutant SH2 domain with reduced capacity for phosphotyrosine binding. Previous experiments in our lab have shown that this construct and the wild type kinase have identical phosphorylation kinetics toward single-site substrates (Amarasinghe and Rosen 2005). Due to sample instability and aggregation at high concentrations, we could measure only k_{cat}/K_M for the AD proteins. All kinase reactions were performed at least in duplicate and at least on two different occasions. Kinetic parameters are reported as an average and standard error of these measurements.

GEF assays. We monitored release of N-methylanthraniloyl-GMPPNP (Mant-GMPPNP) from human Rac1 through decreasing Mant fluorescence as described (Aghazadeh, Lowry et al. 2000), except buffers contained 10 % (w/v) glycerol. Protein concentrations used in the assays: Rac1, 100 nM; AD/ Δ A-D proteins, 50 μ M; CADPZ proteins, 20 μ M. We fit fluorescence time courses to a single exponential decay, $F(t) = \Delta F \cdot \exp(-R \cdot t) + F_0$, where ΔF is total fluorescence decrease, F_0 is the equilibrium fluorescence intensity and R is decay rate constant. Under conditions used, the decay rate constant, R , scales linearly with GEF concentration (not shown). Normalized GEF activity of each protein was obtained from:

$$\text{Normalized GEF activity} = \frac{R_i - R_{control}}{R_{open(i)} - R_{control}} \quad (2)$$

$R_{control}$ is the rate constant without enzyme and $i = AD, AD_{Y174F}, AD_{K208A}, CADPZ, CADPZ_{K208A}$ and $open(i) = \Delta A-D, \Delta A-D, \Delta A-D_{K208A}, CADPZ_{Y174D}, CADPZ_{Y174D K208A}$ respectively. Normalization eliminates effects due to the inherently different activities of $\Delta A-D$ and $\Delta A-D_{K208A}$ or activities of $CADPZ_{Y174D}$ and $CADPZ_{Y174D K208A}$ owing to the K208A mutation in the GEF active site.

Chapter 3 Theoretical Assessment of Parameterization of Four-state Equilibrium Using NMR

Introduction

Allostery refers to the responsiveness/coupling of protein activities to perturbations remote from the active site. Coupled equilibria in the form of allostery are prevalent throughout biology. More and more evidence suggests that allosteric proteins possess intrinsic predisposed equilibria (intrinsic equilibria), which dictate their activities (Kern and Zuiderweg 2003). Allosteric regulators modulate energetic landscapes of intrinsic equilibria through non-covalent and covalent modifications (referred to as the allosteric process hereafter). It is pivotal to decipher coupling strengths built into allosteric systems to fully understand allostery.

Traditionally, allosteric coupling strengths have been determined indirectly by measuring changes in steady-state catalytic activity of an enzyme (Brennan, Christianson et al. 1995; Huang, Shiva et al. 2005; Gladwin and Kim-Shapiro 2008) or ligand binding affinity change of a receptor (Kolb and Weber 1975; Kolb and Weber 1975; Cooper, McAlpine et al. 1994; Popovych, Sun et al. 2006; Muralidhara, Negi et al. 2007) free and when saturated by allosteric regulators. This indirect approach is limited to systems in which there are assayable enzyme activities or binding activities, and full saturation of the

allosteric ligands is achievable. In many systems, however, full saturation is difficult to achieve. For example, in a system where the allosteric regulator is ligand that binds *in trans*, but has low solubility relative to its binding affinity towards the receptor; if the allosteric regulator is *in cis*, the intramolecular binding affinity can be limited. The allosteric ligand-free states can be obtained for both examples as long as the free receptor or the truncated proteins behave well, but the ligand-saturated states cannot be obtained. Coupling of an allosteric process with an intrinsic equilibrium gives rise to a closed four-state equilibrium (Fig. 3-1a). Parameterization of kinetics and thermodynamics of the equilibrium, if possible, provides a direct way to characterize allosteric coupling strengths independent of an assayable activity or experimentally achievable saturation. Developing techniques to solve four-state equilibria via direct parameterization is the focus of this study. One potential technique to serve this purpose is the NMR Carr-Purcell-Meiboom-Gill (CPMG) relaxation dispersion analysis due to its uniqueness in simultaneously determining kinetics and thermodynamics of conformational/chemical exchanges (Palmer, Kroenke et al. 2001).

CPMG has recently been applied to characterize microsecond-millisecond (μ s-ms) time scale dynamics in proteins (Loria, Rance et al. 1999; Loria, Rance et al. 1999; Mulder, Skrynnikov et al. 2001; Palmer, Kroenke et al. 2001; Skrynnikov, Mulder et al. 2001; Tollinger, Skrynnikov et al. 2001; Korzhnev, Kloiber et al. 2004; Korzhnev, Neudecker et al. 2005; Lundstrom, Vallurupalli et

al. 2007). The underlying principle of this technique is that dynamics at μs -ms time scales have extra contributions (termed R_{ex}) to transverse relaxation rate constants, R_2 , via magnetization dephasing (Akke 2002). The spin-echo elements in CPMG pulse sequences can refocus magnetization dephased due to chemical exchange. The refocusing power, and thus the degree to which exchange dynamics influence relaxation, is dictated by the frequency of spin-echo elements (ν_{CPMG}). R_2 values measured as a function of ν_{CPMG} give rise to a relaxation dispersion curve, the profile of which is determined by the kinetics and thermodynamics of chemical exchange, and by the isotropic chemical shifts of the interconverting states.

CPMG can be routinely used to characterize two-state chemical exchange (Kovrigin, Kempf et al. 2006). There are only a few examples dealing with systems undergoing 3-state exchange (Grey, Wang et al. 2003; Tolkmachev, Xu et al. 2003; Korzhnev, Salvatella et al. 2004; Korzhnev, Neudecker et al. 2005; Korzhnev, Bezsonova et al. 2006; Sugase, Dyson et al. 2007). Solving 3-state equilibria using CPMG measurement requires either many redundant measurements (Korzhnev, Neudecker et al. 2005) or orthogonal perturbations such as temperature variations (Grey, Wang et al. 2003; Korzhnev, Salvatella et al. 2004; Korzhnev, Bezsonova et al. 2006) and ligand titrations (Tolkmachev, Xu et al. 2003; Sugase, Dyson et al. 2007). However, it is an uncharted territory to solve four-state equilibria using CPMG techniques. Here we assess the feasibility

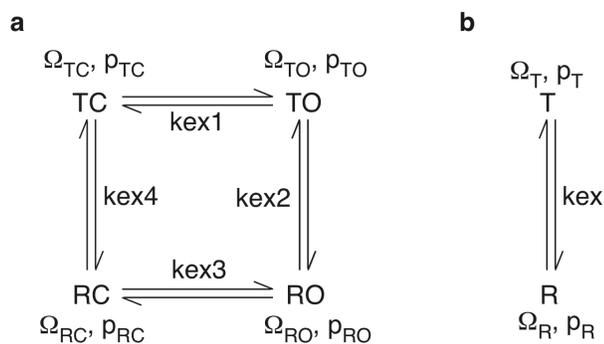


Figure 3-1. The four-state and two-state models. **(a)** A model depicts a four-state model derived from two dynamic processes. The fractional populations are p_{TC} , p_{TO} , p_{RO} and p_{RC} for TC, TO, RO and RC, respectively. $kex1$, $kex2$, $kex3$ and $kex4$ are the sums of the corresponding first order forward and reverse rate constants. For each spin, there are four chemical shifts Ω_{TC} , Ω_{TO} , Ω_{RO} and Ω_{RC} for states TC, TO, RO and RC, respectively. **(b)** A model describes a conformational equilibrium between states T and R. The fractional populations are p_T and p_R for states T and R, respectively. kex , is the sum of the first order forward and reverse rate constants. For each spin there are two chemical shifts Ω_T and Ω_R for states T and R, respectively.

of parameterizing four-state equilibria using CPMG measurements via computer simulation.

In this study, I first establish a strategy for solving four-state equilibria using test parameter sets. Independent determination of one of the least populated edges of a four-state equilibrium (Fig. 3-1a) and incorporation of chemical shift information are required to solve the four-state equilibrium. To guide experimental design, the effects of more measurements on the robustness of the fitted parameters using MC simulation are evaluated.

Results and Discussion

The four-state equilibrium model in AD

Allosteric systems possess pre-equilibria between two or more states. Coupling of allosteric regulators to these systems is through modulation of the dynamic landscape of the pre-equilibrium upon regulator association. Therefore there are minimally two processes in these allosteric systems: the pre-equilibria and the allosteric regulator binding. For example, the μ s-ms timescale dynamics of an autoinhibited Dbl homology (DH) domain construct of Vav1 (a guanine nucleotide exchange factor enzyme) was investigated. In this system catalytic activity is inhibited by direct contact between the DH active site and a helix derived from the adjacent “acid region” (Li, Martins et al. 2008). A Vav1 construct consisting of the inhibitory acidic helix and DH domain (hereafter referred to as AD) has two distinct dynamic processes: one intrinsic to the DH

domain and the other involving interactions between the inhibitory helix and the DH domain (Li, Martins et al. 2008). Following the nomenclature of the MWC model for allostery (Monod, Wyman et al. 1965), the states of the intrinsic process are referred to as T and R without implications toward the functions of the two states. The second process, involving fluctuations between the helix bound (closed) and helix dissociated (open) states of the system is referred to as the CO process.

The combination of the two processes gives rise to a closed (as distinct from linear) four-state equilibrium (Fig. 3-1a). It is assumed that the physical transitions of the intrinsic and CO processes happen instantaneously, and that diagonal transitions involving changes in both processes ($TC \leftrightarrow RO$, $RC \leftrightarrow TO$) are too rare to be observed (Woessner 1961). In Figure 3-1a, the left and right vertical edges are referred as the closed and open edges, respectively; the upper and lower horizontal edges are referred as the T and R edges, respectively. This four-state model is general to all allosteric systems involving two dynamic processes. This study is to establish a strategy to solve four-state equilibria in these allosteric systems using NMR Carr-Purcell-Meiboom-Gill (CPMG) measurements and thereby to characterize allosteric coupling strength in these systems.

Global solution determination

For curve fitting of experimental data, target functions are normally established for the data to be analyzed based on the mathematical relationships between the data and the adjustable parameters. Target functions are minimized to determine the values of the adjustable parameters. Target functions derived from noise-free data assume zero when the adjustable parameters are at their true values (Red dashed curves in Fig. 3-2). Random noise modulates the data, and consequently, the landscapes of the target function surfaces. Notably the noise-incorporated target functions deviate from noise-free ones such that the noise-incorporated target functions no longer assume zero and their minima also deviate from the true values of the adjustable parameters (Fig. 3-2).

Neither the noise-free target functions nor the real values of the adjustable parameters are obtainable in real situations. The assumption for most curve fitting is that noise-incorporated target functions approximate the noise-free ones well enough at least in the vicinity of the true values of the adjustable parameters and the deepest minima correspond to the approximate values of the adjustable parameters, i.e. the global solution. Whether this assumption holds is dependent on the smoothness of the target function surface. The most ideal scenario for curve fitting in general is that the global solution resides in a valley of the target function surface, where each point on the surface has a monotonic descending path towards the global solution and the valley covers the whole of physically

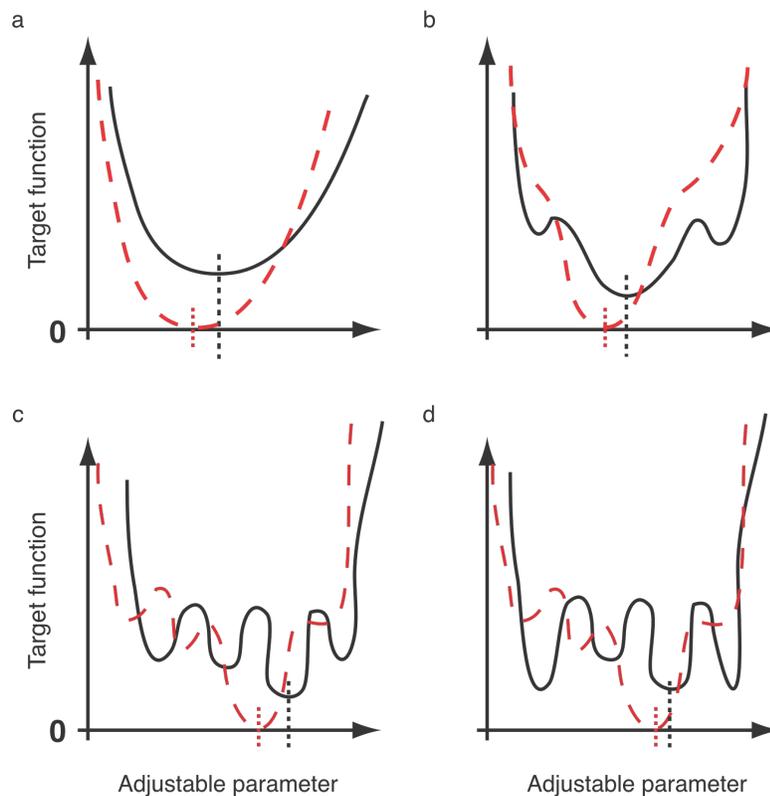


Figure 3-2. Hypothetical noise-incorporated (black solid) and noise-free (red dashed) target function surfaces. The x-axis covers the whole physically meaningful range. The y-axis is the target function and the intercept value of the target function is 0. When there is only one adjustable parameter, the target function is actually a curve. When there are two or more adjustable parameters, the target function is a surface. The target function is always referred to as a surface for convenience purpose. Noise-free target function (red curves) is zero when the adjustable parameter assumes the real values (marked by red vertical dotted lines). The red vertical dotted lines represent the real value of the adjustable parameter. The presence of noise will modulate the target function surfaces to assume nonzero minima. Among the values that correspond to minima on the target function surface derived from noise-incorporated data, the one that is closest to the real value is the global solution (marked by the black vertical dotted lines) and the corresponding minimum is the global minimum. (a) There is still one minimum with the presence of noise. (b) Noise renders two shallower local minima but the global minimum cover the majority of the physically meaningful range of the adjustable parameter. (c) Noise modulates the target function surface such that the global minimum is still the deepest but only covers a portion of the physically meaningful range of the adjustable parameter. (d) Noise alters the target function surface such that the global minimum is no longer the deepest.

meaningful parameter space (Fig. 3-2a). In this case, any algorithm will lead to the global solution. This ideal scenario normally only occurs to curve fitting of simple models where there are only a few adjustable parameters and there is little correlation among them. For example, curve fitting of a binding isotherm with a single dissociation constant is close to this ideal scenario. When the functional forms become a little more complicated, there are more adjustable parameters. There may be local minima on the target function surface but the global solution still resides in the deepest well, which covers the majority of the physically meaningful range of the adjustable parameter (Fig. 3-2b). It is not difficult to find the global solution for this scenario, either. When the functional form becomes even more complicated, there are even more adjustable parameters and these parameters are possibly correlated. There may be more local minima and the global solution still resides in the deepest well, which only covers a portion of the physically meaningful range of the adjustable parameter (Fig. 3-2c). This scenario is more challenging for curve fitting and sophisticated algorithms are required to obtain the global solution. Nevertheless, the minima in the deepest well for the three scenarios correspond to the global solutions and therefore the set of parameters that gives the lowest target function is indeed the global solution. For some very complicated functional forms, there are a lot of adjustable parameters and these parameters share complex inter-parameter correlations. Even low level of noise can modulate the target function so much that the global

solution no longer resides in the deepest well and therefore the set of parameters that gives the lowest target function isn't the global solution any more (Fig. 3-2d). For the last scenario, more information is required to determine the global solutions. In summary, target function surface mapping is a good start for evaluating whether it is feasible to characterize complicated models, for example to parameterize four-state equilibria using CPMG measurements in this Chapter.

χ^2 surface mapping

To theoretically assess in what parameter space it is feasible to parameterize four-state equilibria using CPMG measurements, I first constructed a benchmark based on experimental results in Chapter 4. The four-state equilibrium of AD at 5 °C was solved using the strategy to be described in this Chapter. Kinetic and thermodynamic parameters of the solved four-state equilibrium and chemical shifts of the eight spins of the best CPMG data were selected as the benchmark. Noise-free 600 MHz and 800 MHz CPMG data were synthesized assuming a four-state equilibrium for this benchmark, into which 4% random noise was incorporated (See Materials and Methods section for details of the synthesized data). The target function for CPMG data is a χ^2 function defined as:

$$\chi^2(\zeta) = \sum \frac{(R_{2,\text{eff}}^{\text{calc}}(\zeta) - R_{2,\text{eff}})^2}{(4\% * R_{2,\text{eff}})^2} \quad (1)$$

where $R_{2,\text{eff}}$ is data to be fitted, $R_{2,\text{eff}}^{\text{calc}}(\zeta)$ is calculated R_2 given a set of adjustable parameters during the process of curve fitting, $\zeta = (x_1, \dots, x_{\text{npa}})$ represents the set of adjustable parameters and the summation is over all relaxation data points.

To evaluate the χ^2 surface, gradient descent-based optimization was performed from multiple initial points (See Materials and Methods section for details of the initial points), each of which leads to one solution, i.e. a minimum on the χ^2 surface. In order to assess similarity among the solutions, which is dictated by the actual χ^2 surface, logarithms of averaged normalized distances (LAND) between any pair of the solutions are calculated according to:

$$\text{LAND}(S1 \leftrightarrow S2) = \max(-1.5, \log_{10} \sqrt{\frac{\sum_{i=1}^n \frac{(S1(i) - S2(i))^2}{(|S1(i)| + |S2(i)|)^2}}{n}}) \quad (2)$$

in which, S1 and S2, each being a solution with n-dimension and each element corresponds to one adjustable parameter. Any LAND below -1.5, including diagonal distances, is set to -1.5, which corresponds to an upper limit of 6.3% deviation per adjustable parameter between two solutions. LAND of a 7-dimensional vector containing 4 kex values and 3 populations and of a 24-dimensional vector containing 24 chemical shifts is plotted in Figures 3-3a. LAND of a 24-dimensional vector containing 24 chemical shifts is shown in Figure 3-3b. There are barely any dark blue clusters in either of the plots, suggesting that most of the initial points lead to distinct minima (Figs. 3-3a and 3-

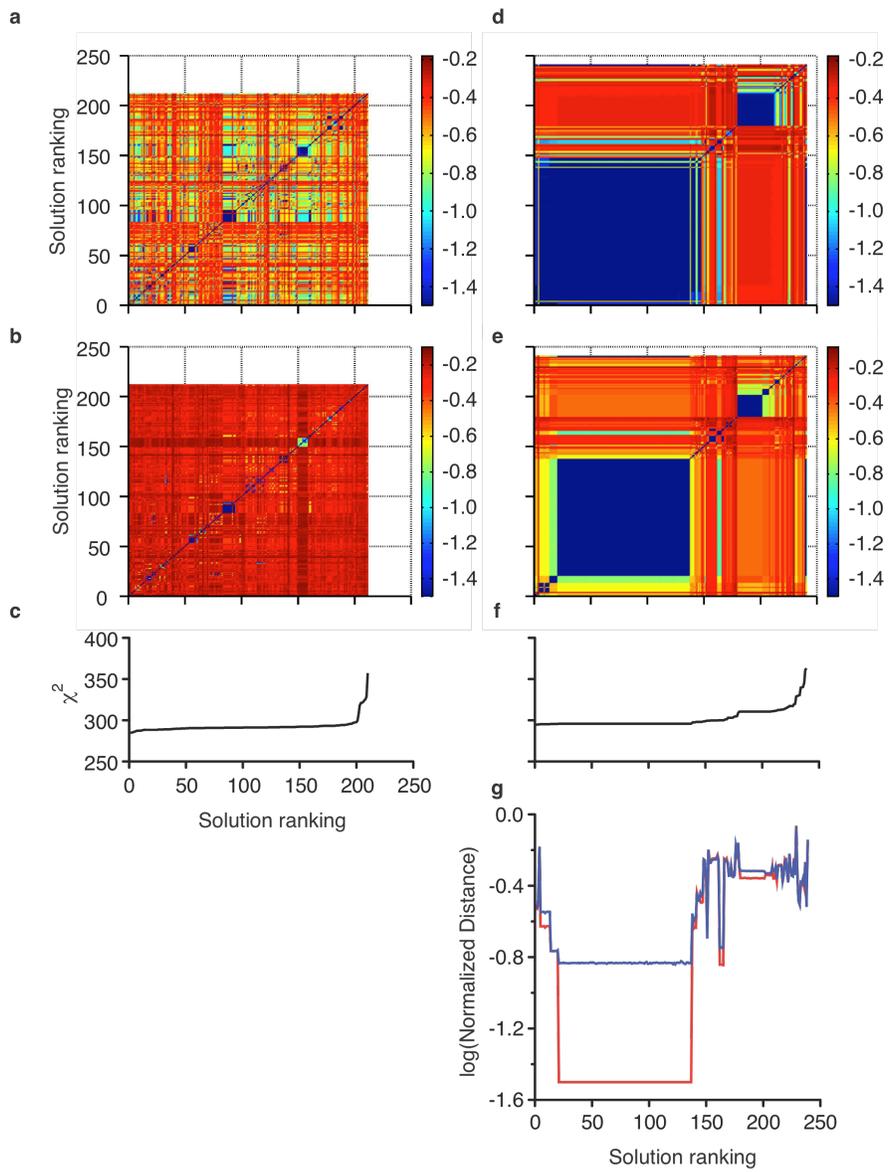


Figure 3-3. Curve fitting results of synthesized CPMG data. Logarithms of averaged normalized pair wise distances (LAND) of the 7-dimensional kinetic/thermodynamic parameter vectors and of the 24-dimensional chemical shift vectors of all solutions in curve fitting without constraints from the two-state proteins are shown in **(a)** and **(b)**, respectively. The corresponding χ^2 values are plotted in **(c)**. LAND of the 5-dimensional kinetic/thermodynamic parameter vectors and of the 16-dimensional chemical shift vectors of all solutions in curve fitting with constraints from the two-state proteins are shown in **(d)** and **(e)**, respectively. The corresponding χ^2 values are plotted in **(f)**. The last two rows and columns in **(d)** and **(e)** corresponds to the input vector and the solution obtained by gradient descent search from the input vector. The columns corresponding the input vector in **(d)** and **(e)** are combined and plotted as the blue curve in **(g)**. The columns corresponding the solution obtained by gradient descent search from the input vector in **(d)** and **(e)** are combined and plotted as the red curve in **(g)**. The rankings of solutions in all panels are based on the ascending order of their corresponding χ^2 except for the last two rows and/or columns in **(d-f)**. Panels **(a-c)** share x-axis. Panels **(d-g)** share x-axis. Panels **(d)**, **(e)** and **(f)** share y-axis with panels **(a)**, **(b)** and **(c)**, respectively.

3b). In addition, most of the solutions have comparable χ^2 values (Fig. 3-3c), suggesting that the χ^2 surface is of high roughness and fits the abovementioned worst case scenario (Fig. 3-2d). Since solving a simpler system, a linear three-state model, requires more orthogonal measurements (Korzhnev, Salvatella et al. 2004; Korzhnev, Neudecker et al. 2005; Korzhnev, Bezsonova et al. 2006; Neudecker, Korzhnev et al. 2006), this behavior is expected for a four-state equilibrium.

Effects of incorporation of orthogonal information

Given the roughness of the χ^2 surface (Fig. 3-3a and 3-3b), orthogonal information is required to parameterize the four-state equilibrium. Any information that provides constraints on the adjustable parameters without introducing new adjustable parameters will help to smooth the χ^2 surface. Measurements that can exogenously determine some parameters contributing marginally to relaxation dispersion in the four-state systems are of the greatest benefit. Such information is readily available. For instance, the two-state equilibrium of a free receptor represents one edge of the four-state equilibrium. Similarly in AD, phosphorylation mimetic mutations (e.g. K208E) strongly disfavor the binding of the helix to the DH domain and the system is reduced to the open edge of the four-state equilibrium in AD. This edge makes only a minor contribution to relaxation dispersion of AD due to the low population of the open state (i.e. TO + RO is small) (Li, Martins et al. 2008). The edge manifested in

AD_{K208E} or free receptor, denoted as the open edge in Figure 3-1a, can be well determined in these two-state systems. Fitted kinetic, thermodynamic and chemical shift parameters from two-state systems can be used as constraints in curve fitting of four-state systems, AD or partially saturated receptors.

Curve fitting from multiple initial points of the same set of CPMG data were carried out by constraining adjustable parameters of the open edge to mimic inputs from a two-state protein. LAND of the solutions (a 5-dimension vector: 3 k_{ex} values, 2 populations and a 16-dimension vector: 16 chemical shifts) are plotted in Figures 3-3d and 3-3e. There is one major blue cluster of low χ^2 in kinetic and thermodynamic parameter plot (Fig. 3-3d). This cluster diverges into a large cluster and some minor ones in the chemical shift plot (Fig. 3-3e), suggesting that chemical shifts suffer from greater degeneracy than kinetic and thermodynamic parameters. The large cluster in the chemical shift plot has the shortest distance from the input vector (Fig. 3-3f), suggesting that it is the global solution.

Interestingly, the corresponding fitted parameters resulting from gradient descent search from the input vector coincides with the large cluster in the chemical shift plot, i.e. the global solution (red curve in Fig. 3-3f). All Monte Carlo simulations in the following sections are based on this observation.

Global solution determination in empirical application

It is worth noting that there are 20 solutions of χ^2 (between 294.9 and 295.6) slightly lower than the global solution (295.9) (Figure 3e-3f). In contrast to curve fitting of simpler models (Fig. 3-2a-c), the χ^2 value alone is not sufficient to determine the global solution for four-state equilibria (Fig. 3-2d). In addition, foreknowledge of the global solution would be unavailable in experimental applications. Further orthogonal information is needed to pinpoint the global solution. Such orthogonal information is also available, in the form of the chemical shift differences (CSDs) between the two-state proteins (AD_{K208E} or free receptors) and the four-state proteins (AD or partially saturated receptors) (defined as CSD^{HSQC}). CSD^{HSQC} is calculated according to:

$$\text{CSD}^{\text{HSQC}} = \left| \left(p_{\text{TC}} * \Omega_{\text{TC}} + p_{\text{TO}} * \Omega_{\text{TO}} + p_{\text{RO}} * \Omega_{\text{RO}} + p_{\text{RC}} * \Omega_{\text{RC}} \right) - \frac{p_{\text{TO}} * \Omega_{\text{TO}} + p_{\text{RO}} * \Omega_{\text{RO}}}{p_{\text{TO}} + p_{\text{RO}}} \right| \quad (3)$$

in which parameters on the right hand side were their input values. The calculated chemical shift differences (CSD^{CPMG}) can be obtained according to the same equation using fitted population and chemical shifts for each solution.

To determine which solution is the global solution, i.e. the one recapitulates the input parameters the best, the slopes and the coefficients of determination (r^2) of the linear correlation between CSD^{HSQC} and CSD^{CPMG} of all solutions are determined and plotted against each other in Figure 3-4a.

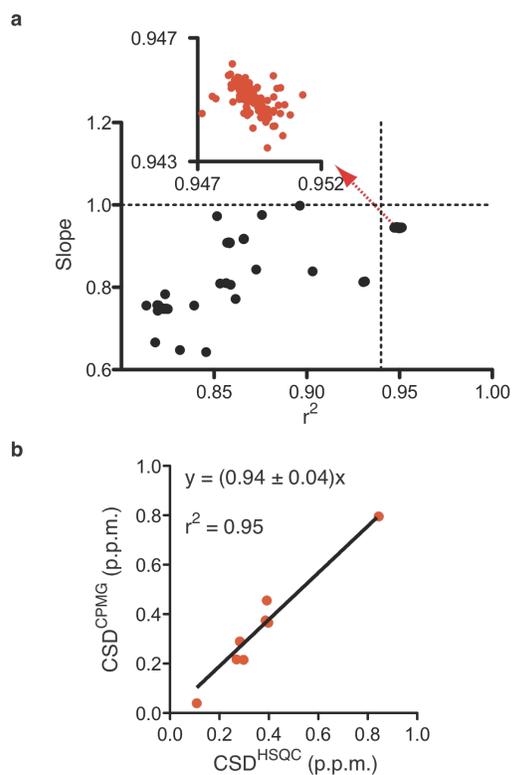


Figure 3-4. Determining the global solution using chemical shift differences. The slopes and the coefficients of determination (r^2) of the linear correlations of chemical shift differences calculated from the solutions (CSD^{CPMG}) with chemical shift differences obtained from input parameter (CSD^{HSQC}) to mimic experimental measurement in empirical application are plotted against each other in (a). 56 data points are out of the axis limits. The dotted vertical line marks 0.94 on x-axis and the dotted horizontal line marks unit on y-axis. There are 117 data points with r^2 larger than 0.94 (inset of (a)). The plot of CSD^{CPMG} from the global solution (randomly picked from the 117 solutions in the inset of (a)) with CSD^{HSQC} is shown in (b).

Remarkably 117 solutions have r^2 values beyond 0.94 (marked by the vertical dotted line) (0.947~0.952) and slopes close to unity (marked by the horizontal dotted line) (0.943~0.947) (inset of Fig. 3-4a). The 117 solutions coincide perfectly with the big cluster of chemical shift plot, i.e. the global solution (Figure 3-3e). The plot of CSD^{HSQC} versus CSD^{CPMG} of the global solution is shown in Figure 3-4b. In summary, this chemical shift difference comparison assay can be used to determine the global solutions. It is worth noting that this approach, however, doesn't work for slow exchange scenarios.

Robustness of the global solution

To evaluate the robustness of the global solution, one thousand runs of Monte Carlo (MC) simulation were carried out. In each run, a set of 4% random noise was incorporated into the noise-free data set. Ideally the strategy established above should be employed to find the global solution for each run of MC simulation. Based on the observation that the solution obtained through from the input parameter vector is the global solution (Fig. 3-3g), the solution for each run of MC simulation was obtained through gradient descent-base search from the input vector for MC simulation, i.e. the global solution of the benchmark. The fitted kinetic and thermodynamic parameters of the MC simulations and the global solution are shown in Table 3-1. Among the five adjustable kinetic/thermodynamic parameters, k_{ex4} and p_{TC} are accurately and precisely determined; k_{ex1} and p_{RC} are reasonably but less well-defined than k_{ex4} and p_{TC} ;

Table 3-1 Robustness of the global solution towards random noise

	Input	Global Solution	MC simulation
kex1	6.5	6.0 ± 0.3	6.3 ± 0.5
kex3	6.5	6.8 ± 0.9	6.5 ± 1.8
kex4	6.5	6.7 ± 0.2	6.4 ± 0.5
p _{TC}	0.85	0.84 ± 0.01	0.84 ± 0.02
p _{RC}	0.05	0.043 ± 0.017	0.059 ± 0.023

kex3 is not well determined. Nevertheless the MC simulation results are statistically indistinguishable from the global solution Table 3-1.

Feasibility in major parameter space

In order to comprehensively survey in what parameter space it is feasible to solve four-state equilibria, more test parameter sets were constructed for a common scenario, in which one of the four states is majorly populated (≥ 0.8) (Kovrigin, Kempf et al. 2006). In terms of thermodynamic parameters, $p_C (= p_{TC} + p_{RC}) = 0.9$; to sample different degrees of skewedness in the C edge, three situations in (p_{TC}, p_{RC}) are considered: (0.88, 0.02), (0.85, 0.05) and (0.80, 0.10) (Figure 3-5a); to sample different degrees of skewedness in the O edge, within each (p_{TC}, p_{RC}) , 13 cases of (p_{TO}, p_{RO}) are considered: (0.005, 0.095), (0.01, 0.09), (0.015, 0.085), (0.02, 0.08), (0.03, 0.07), (0.04, 0.06), (0.05, 0.05), (0.06, 0.04), (0.07, 0.03), (0.08, 0.02), (0.085, 0.015), (0.09, 0.01) and (0.095, 0.005) (Figure 3-5b). For each population combination (total $3 \times 13 = 39$), each of $\ln(k_{ex1})$ to $\ln(k_{ex4})$ takes either 6.5 ($\sim 665 \text{ s}^{-1}$) or 7.5 ($\sim 1808 \text{ s}^{-1}$) and combination of four kexs gives rise to 16 cases ($16 = 2^4$) (Figure 3-5c). There are 624 test parameter sets in total. For each parameter set, noise-free 600 MHz and 800 MHz CPMG data were generated for the same eight spins used above. MC simulation was then used to assess how well each parameter could be defined in the presence of 4% random noise.

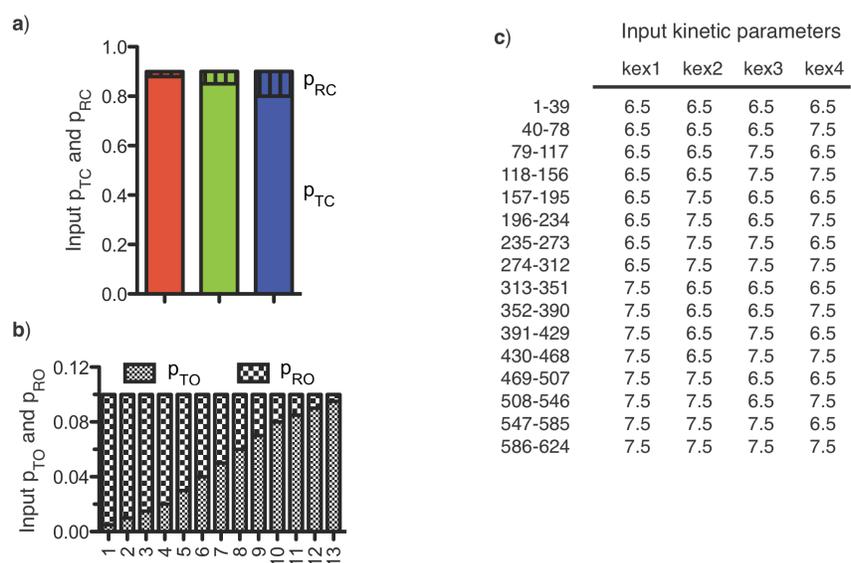


Figure 3-5. Input kinetic and thermodynamic parameters in parameter space survey. p_C ($= p_{TC} + p_{RC}$) is 0.9; three p_{TC}/p_{RC} ratios are considered: 0.88/0.02 (red bar), 0.85/0.05 (green bar) and 0.80/0.10 (blue bar) (a); 13 p_{TO}/p_{RO} ratios are considered: 0.005/0.095, 0.01/0.09, 0.015/0.085, 0.02/0.08, 0.03/0.07, 0.04/0.06, 0.05/0.05, 0.06/0.04, 0.07/0.03, 0.08/0.02, 0.085/0.015, 0.09/0.01 and 0.095/0.005 (b); each of $\ln(kex1)$ to $\ln(kex4)$ takes either 6.5 ($\sim 665 \text{ s}^{-1}$) or 7.5 ($\sim 1808 \text{ s}^{-1}$) and combination of four kexs give rises to 16 cases (2^4) (c).

One hundred runs of MC simulation were carried out for each parameter set through gradient descent-based search from the corresponding input parameter set. The means and standard deviations of the normalized fitted kinetic/thermodynamic parameters are shown in Figures 3-6. As shown in Figure 3-6a, k_{ex1} is reasonably determined when p_{TO} is high (later part of each of the 48 colored sectors), presumably because in this situation the T edge makes a larger contribution to relaxation. In contrast, Fig. 3-6b shows that k_{ex3} is reasonably determined when p_{RO} is high (early part of each of the 48 colored sectors) due to larger contribution to relaxation from the R edge instead. Owing to the opposite responses to alterations in p_{TO}/p_{RO} , k_{ex1} and k_{ex3} cannot be simultaneously well determined (Figs. 3-6a and 3-6b). The last kinetic parameter, k_{ex4} is very well defined when the C edge is less skewed, i.e. $p_{TC}/p_{RC} = 0.85/0.05$ or $0.8/0.1$ (green and blue bars in Fig. 3-5f). Interestingly, both the accuracy and precision of the fitted k_{ex4} are compromised when $k_{ex1} = 7.5$, $k_{ex4} = 6.5$, $p_{TC}/p_{RC} = 0.88/0.02$ and p_{TO}/p_{RO} is low (9^{th} , 11^{th} , 13^{th} and 15^{th} red bars in Fig. 3-6c). This is because the T edge replaces the C edge as the relaxation dominating edge. In summary, for a k_{ex} to be well determined, its corresponding edge has to be highly populated, which in turn means that that edge contributes more significantly to relaxation.

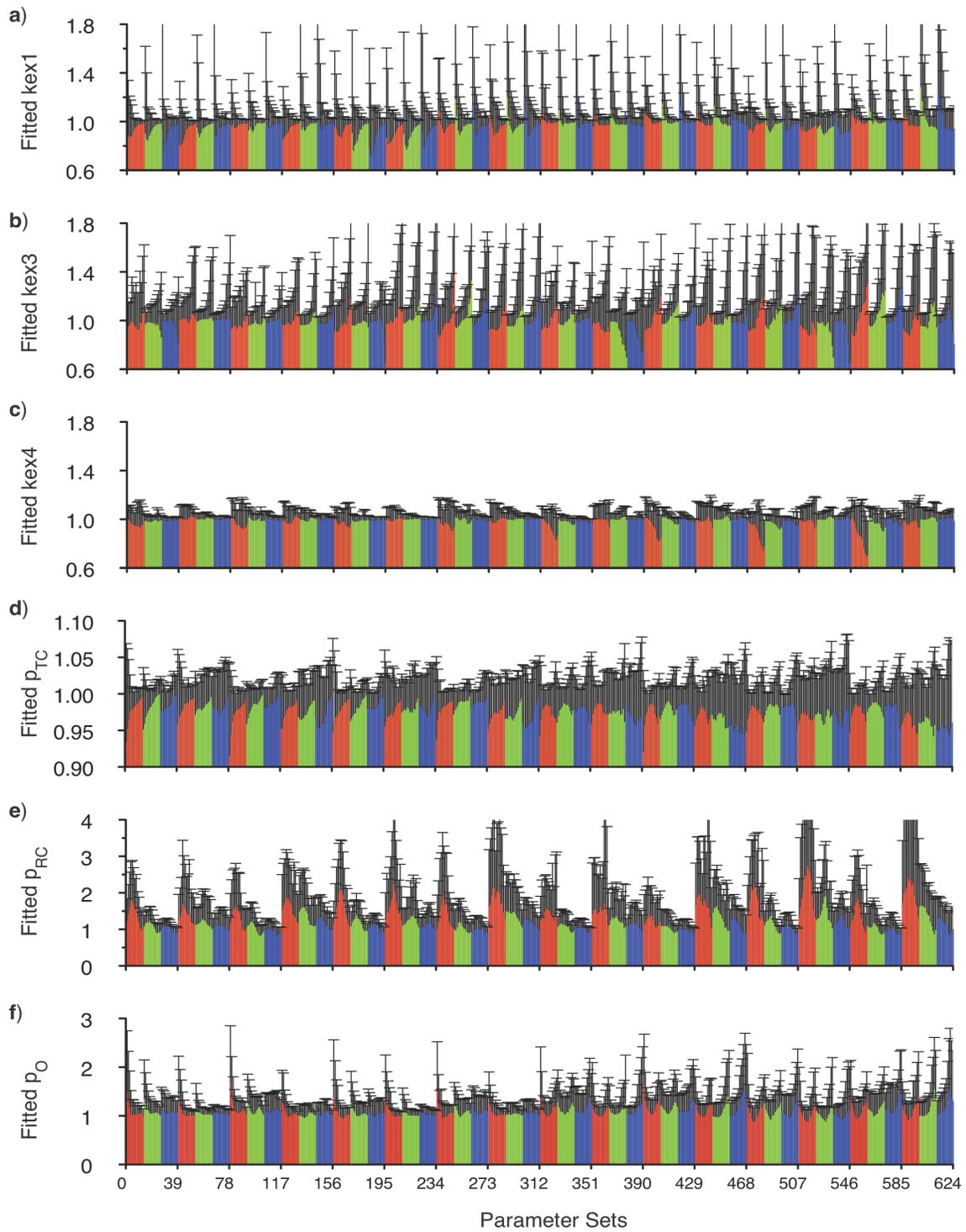


Figure 3-6. Fitted kinetic and thermodynamic parameters in parameter space survey. Fitted k_{ex1} (a), k_{ex3} (b), k_{ex4} (c), p_{TC} (d), p_{RC} (e) and the ultimate p_O (f) are shown. Kinetic parameters are in natural logarithms. Each parameter is normalized to its input value. The error bars are the standard deviations of the normalized fitted parameters. In each panel, there are 16 red and 16 green and the 16 blue sectors and the color of each sector is coding for p_{TC}/p_{RC} ratio as in Figure 3-5a. Each of the 48 (16 red + 16 green + 16 blue) sectors contains 13 individual bars. The 13 bars sequentially correspond to the 13 p_{TO}/p_{RO} ratios in Figure 3-5b. The horizontal sequence from left to right of 16 sectors of common color is coding the vertical sequence from top to bottom of kinetic parameter combinations in Figure 3-5c. Totally there are 624 individual bars in each panel, corresponding to 624 parameter sets.

As far as adjustable populations are concerned, p_{TC} is very well determined since it is the major species (Fig. 3-6d); p_{RC} is better determined when its input value is high and it tends to be over-estimated when its input value is low (Fig. 3-6e). Since the thermodynamics of the open edge are fixed, the fitted values of p_{TC} and p_{RC} dictate the accuracy and precision of the calculated open populations:

$$p_O = p_{TO} + p_{RO} = 1 - (p_{TC} + p_{RC}) \quad (4)$$

The calculated open populations are plotted in Figure 3-6f. Open populations tend to be more affected when k_{ex1} is larger (compare the first half and the second half of Figure 3-6f). The largest uncertainties arise from cases where p_{TO} is also low. Nevertheless populations of four-state equilibria can be reasonably characterized in most of the test cases. In general, a parameter can be well defined only if it contributes significantly to relaxation.

Improvement upon addition of more measurements

It is instructive to know how much additional measurements can improve the fitted parameters. Given the current availability of 900 MHz instruments, there is an option of either collecting data at this additional field or repeating the measurements at 600 MHz and 800 MHz. To estimate the improvement that these additions could bring to the fitted parameters, the parameter set used in χ^2 surface mapping was used as a benchmark here. The calculated CPMG data were

either duplicated to mimic repeating the measurements at 600 MHz and 800 MHz or incorporated with 900 MHz data to mimic addition of data at a higher field. The two resulting noise-free data sets were then incorporated with 1000 sets of 4% random noise for 1000 runs of MC simulation. The fitted kinetic and thermodynamic parameters were normalized to the input values, and their means and standard deviations are shown in Figure 3-6. Repeating the measurements at 600 MHz and 800 MHz fields and incorporation of 900 MHz data each improve both the accuracy and the precision of fitted parameters (Fig. 3-6). The two better determined kinetic parameters, k_{ex1} and k_{ex4} , show the largest relative improvement. For example the standard deviation of k_{ex1} dropped more than 60% when 900 MHz data were incorporated. The most poorly determined two parameters, k_{ex3} and p_{RC} , remain poorly determined with either addition, although they do show slight improvement in their precision upon these treatments. Overall, incorporation of 900 MHz data is superior to repeating measurement on the two lower fields with this parameter set, improving precision more than the latter in 4 out of the 5 parameters.

Using a suite of six NMR relaxation dispersion experiments on ^1H - ^{15}N amide moieties, Kay and coworkers have solved a linear three-state model robustly (Korzhnev, Neudecker et al. 2005; Neudecker, Korzhnev et al. 2006). Assessment of effects of incorporation of the six experiments is beyond the scope of the current study. The advantage of methyl ^{13}C -based measurements over

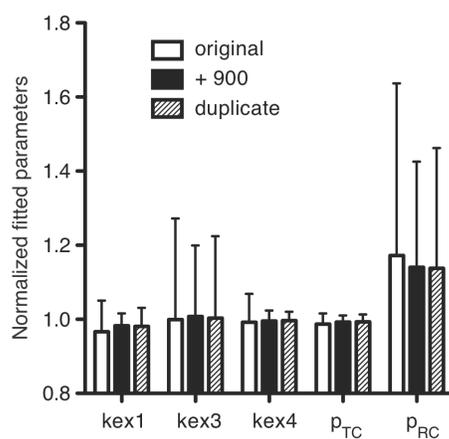


Figure 3-7. Effects of more measurements on fitted kinetic and thermodynamic parameters. Fitted kex1, kex3, kex4, p_{TC} and p_{RC} either of original 600 MHz and 800 MHz data (empty bars) or of original data incorporated with 900 MHz data (solid black bars) or of duplicate of the original data (striped bars) are shown. Each parameter is normalized to its input value. The error bars are the standard deviations of the normalized parameters.

amide ^1H - ^{15}N based ones is the potential of high signal to noise ratio, which is crucial for solving four-state equilibria.

Conclusions

Allosteric systems can be described in thermodynamic terms by a four-state box derived from the pre-equilibrium and allosteric effector binding (Fig. 3-1). It is the coupling between the equilibria that enables one process to communicate with the other. It is coupling strengths that dictate the information flow capacity from one process to another. Therefore deciphering coupling strengths built into allosteric proteins is pivotal in fully understanding allostery. In this Chapter, I established a strategy to directly characterize allosteric coupling strengths by parameterizing four-state equilibria using NMR CPMG techniques. This strategy circumvents limitations in conventional way of determining allosteric coupling strengths. In addition, I also addressed the following related questions: in parameter space parameterization of four-state equilibria using current strategy is feasible; how much additional measurements can improve the fitted parameters? Work in this Chapter paves a new avenue for characterizing allosteric coupling strengths, especially in cases involving signaling molecules without assayable readout.

Materials and Methods

Generation of dispersion profiles

Synthetic relaxation dispersion data of a four-state equilibrium were obtained based on the major species (≥ 0.8), TC, using:

$$R_{2,\text{eff}} = -\frac{1}{T} * \ln \frac{M_{\text{TC}}(T)}{M_{\text{TC}}(0)} \quad (5)$$

where $M_{\text{TC}}(T)$ and $M_{\text{TC}}(0)$ are the first element of $\mathbf{M}(T)$ and $\mathbf{M}(0)$ in Equation 4 below:

$$\mathbf{M}(T) = [\exp(\mathbf{A}_-\delta)\exp(\mathbf{A}_+\delta)\exp(\mathbf{A}_-\delta)\exp(\mathbf{A}_+\delta)]^n \mathbf{M}(0) \quad (6)$$

$T = 4n\delta$ with $2n$ the number of spin-echo trains within the constant time period,

$\mathbf{M}(T)$ is the magnetization vector $(M_{\text{TC}}(T), M_{\text{TO}}(T), M_{\text{RO}}(T), M_{\text{RC}}(T))^T$, $\mathbf{M}(0)$ is the initial magnetization vector $(p_{\text{TC}}, p_{\text{TO}}, p_{\text{RO}}, p_{\text{RC}})^T$, \mathbf{A}_\pm is the evolution of magnetization matrix before (\mathbf{A}_-) and after (\mathbf{A}_+) a 180° pulse in a spin-echo train

(Eq. 7)

$$\mathbf{A}_\pm = \begin{bmatrix} -k1 - k7 - R_2 \pm i * \Omega_{\text{TC}} & k2 & 0 & k8 \\ k1 & -k2 - k3 - R_2 \pm i * \Omega_{\text{TO}} & k4 & 0 \\ 0 & k3 & -k4 - k6 - R_2 \pm i * \Omega_{\text{RO}} & k5 \\ k7 & 0 & k6 & -k5 - k8 - R_2 \pm i * \Omega_{\text{RC}} \end{bmatrix} \quad (7)$$

and all other parameters are defined in the legend to Figure 3-1a. Since we were dealing with intermediate and fast exchange regime in CPMG data generation and

curve fitting, i.e. kinetic parameters are far bigger than the intrinsic transverse relaxation rate constants and therefore differences of intrinsic R_2 of different states can be neglected (Grey, Wang et al. 2003), we assume R_2^0 possesses identical value 20 s^{-1} . Chemical shifts of eight methyl groups in AD of the best CPMG data in Chapter 4 were picked as the benchmark. The eight chemical shift vectors (Ω_{TC} , Ω_{TO} , Ω_{RO} , Ω_{RC}), one for each methyl group, in part per million (ppm) were: (0.9, 0.0, 0.0, 0.6), (0.3, 0.0, 0.0, 0.7), (0.45, 0.0, 0.0, -0.15), (0.5, 0.0, 0.0, 0.6), (-0.1, 0.0, 0.65, 0.9), (-0.1, 0.0, 0.6, 0.6), (0.14, 0.0, 0.7, 0.6) and (-0.3, 0.0, 0.5, 0.5). In terms of thermodynamic parameters, $p_C (= p_{TC} + p_{RC}) = 0.9$; three situations in (p_{TC}, p_{RC}) are considered: (0.88, 0.02), (0.85, 0.05) and (0.80, 0.10) (Figure 3-5a); within each (p_{TC}, p_{RC}) , 13 cases of (p_{TO}, p_{RO}) are considered: (0.005, 0.095), (0.01, 0.09), (0.015, 0.085), (0.02, 0.08), (0.03, 0.07), (0.04, 0.06), (0.05, 0.05), (0.06, 0.04), (0.07, 0.03), (0.08, 0.02), (0.085, 0.015), (0.09, 0.01) and (0.095, 0.005) (Figure 3-5b). For each population combination (total $3 \times 13 = 39$), each of $\ln(k_{ex1})$ to $\ln(k_{ex4})$ takes either 6.5 ($\sim 665 \text{ s}^{-1}$) or 7.5 ($\sim 1808 \text{ s}^{-1}$) and combination of four k_{ex} s gives rise to 16 cases ($16 = 2^4$) (Figure 3-5c).

Noise-free 600 MHz and 800 MHz (some also with 900 MHz data) single quantum ^{13}C CPMG data were generated using an in-house written Mathematica script (Appendix 1). The resulting dispersion profiles $R_{2,\text{eff}}(v_{\text{CPMG}})$ included 20 points with v_{CPMG} varying from 50 to 1000 Hz and 50 Hz as the step size. There

were total 16 dispersion profiles for eight resonances at two fields. To mimic incorporation of 900 MHz data, another 8 dispersion profiles, one corresponding to each resonance at 900 MHz, were added to make it 24 curves in total. To mimic repeating this set of measurements at 600 MHz and 800 MHz, the 16 noise-free profiles were duplicated to give 32. Each data point (with the value of $R_{2,\text{eff}}$) of the noise-free dispersion data set was incorporated with a noise term randomly drawn from a Gaussian distribution of mean zero and standard deviation of $4\% * R_{2,\text{eff}}$ to give rise to 4% noise incorporated data. Given the availability of cryogenic probes, perdeuteration labeling and sensitivity enhanced NMR techniques, 4% is a practical noise level for many systems (Neudecker, Korzhnev et al. 2006).

Curve fitting of CPMG data

We used a script modified from the C code provided by Drs. D.M.K. and L.E.K. to globally fit synthesized data by nonlinear least square minimizing the χ^2 functions:

$$\chi^2(\zeta) = \sum \frac{(R_{2,\text{eff}}^{\text{clc}}(\zeta) - R_{2,\text{eff}})^2}{(4\% * R_{2,\text{eff}})^2} \quad (8)$$

where $R_{2,\text{eff}}^{\text{clc}}(\zeta)$ is calculated the same way as $R_{2,\text{eff}}$ given a set of adjustable parameters, $\zeta = (x_1, \dots, x_{\text{npa}})$ represents the set of adjustable parameters and the summation in Eq. 8 is over all relaxation data points.

In the process of curve fitting without constraints from the two-state proteins, we did gradient-descent search from multiple points among which $\ln(k_{ex1})$, $\ln(k_{ex3})$ and $\ln(k_{ex4})$ varied from 5.0 to 8.0 with step size 0.2; p_{TC} varied from 0.5 to 0.78 with step size 0.02; $p_{RC} = 0.8 - p_{TC}$; initial $\ln(k_{ex2})$ equals its input value 7.5; $p_{TO} = 0.14$ (input value: 0.07); $p_{RO} = 0.06$ (input value: 0.03); initial Ω_{RO} equal to the input values; initial Ω_{TC} and initial Ω_{RC} of the first four spins were set to be equal and their values were arbitrarily set according to the relaxation dispersion amplitude; initial Ω_{TC} of the last four spins equal to the corresponding Ω_{TO} s; initial Ω_{RC} of the last four spins equal to the corresponding Ω_{BO} ; initial intrinsic R_2 s were set at their input values.

In the process of curve fitting with constraints from the two-state proteins, we did gradient-descent search from multiple points among which $\ln(k_{ex1})$ and $\ln(k_{ex4})$ varied from 5.0 to 8.0 with step size 0.2; p_{TC} varied from 0.5 to 0.78 with step size 0.02; $p_{RC} = 0.8 - p_{TC}$; the same treatment to chemical shifts as above.

For any two solutions, S1 and S2, each being an n dimension vector and each element corresponding to one adjustable parameter, the distance between them was calculated according to the following equation:

$$\text{LAND}(S1 \leftrightarrow S2) = \max(-1.5, \log_{10} \sqrt{\frac{\sum_{i=1}^n \frac{(S1(i) - S2(i))^2}{(|S1(i)| + |S2(i)|)^2}}{n}}) \quad (9)$$

For any resonance i , the chemical shift difference obtained from HSQC (CSD^{HSQC}) was calculated based the equation below:

$$\text{CSP}^{\text{HSQC}} = \left| (p_{\text{TC}} * \Omega_{\text{TC}} + p_{\text{TO}} * \Omega_{\text{TO}} + p_{\text{RO}} * \Omega_{\text{RO}} + p_{\text{RC}} * \Omega_{\text{RC}}) - \frac{p_{\text{TO}} * \Omega_{\text{TO}} + p_{\text{RO}} * \Omega_{\text{RO}}}{p_{\text{TO}} + p_{\text{RO}}} \right| \quad (10)$$

in which parameters on the right hand side were their input values. For the same resonance, the CSD obtained from CPMG (CSD^{CPMG}) was calculated based the equation below:

$$\text{CSP}^{\text{CPMG}} = \left| (p_{\text{TC}} * \Omega_{\text{TC}} + p_{\text{TO}} * \Omega_{\text{TO}} + p_{\text{RO}} * \Omega_{\text{RO}} + p_{\text{RC}} * \Omega_{\text{RC}}) - \frac{p_{\text{TO}} * \Omega_{\text{TO}} + p_{\text{RO}} * \Omega_{\text{RO}}}{p_{\text{TO}} + p_{\text{RO}}} \right| \quad (11)$$

in which all parameters on the right hand side were their fitted values in the solutions. This approach assumes fast exchange regime. It doesn't work for slow exchange regime.

Monte Carlo Simulation

Monte Carlo (MC) simulation was used to assess the robustness of the fitted parameters toward random noise. In one run of MC simulation, each data point of the noise-free relaxation dispersion data set was incorporated with 4% random noise and the whole set of noise-incorporated CPMG data was subject to gradient descent-based nonlinear least square fitting from the input values of

adjustable parameters.

Chapter 4 Allosteric Coupling Strengths between Two Dynamic Processes

Introduction

Allostery refers to the coupling of a functional process in a protein to a remotely located process. For instance, catalytic activities of enzymes or ligand binding affinities of receptors can be modulated by the association of effector molecules at distal sites (Goodey and Benkovic 2008). Effector molecules encompass any regulatory molecules, including small molecules, proteins or domains *in cis* to the functional process. Allosteric regulation is crucial for life as it is one of the primary forms of information flow within proteins that enables response to external cues (Gunasekaran, Ma et al. 2004). For example, allostery is involved in feedback from downstream products and feedforward from upstream substrates in metabolic pathways (Monod, Changeux et al. 1963; Umbarger 1992). External cues can be either the effector molecules themselves or act on the effector molecules to control the pre-equilibrium (Boehr, McElheny et al. 2006; Boehr and Wright 2008).

A well-known model for allosteric systems is that proteins possess a pre-equilibrium between a T state and an R state and effector molecules modulate the dynamic landscape of the pre-equilibrium (Monod, Wyman et al. 1965; Boehr, Dyson et al. 2006; Boehr and Wright 2008). Allosteric systems can be described

in thermodynamic terms by a four-state box derived from the pre-equilibrium and allosteric effector binding (Fig. 4-1) (Kolb and Weber 1975; Kolb and Weber 1975). It is the coupling between the equilibria that enables one process to communicate with the other. The coupling strengths dictate the information flow capacity from external cues to interior of the cells through allosteric proteins. Therefore deciphering coupling strengths built into allosteric proteins is pivotal in fully understanding allostery and thus in understanding how information flows between molecules in cells. Such understanding also has practical implications, as it could guide the design of allostery-based biosensors (Brennan, Christianson et al. 1995; Kay 1998; Villaverde 2003; Yao, Rosen et al. 2008) and drugs that act through allosteric mechanisms (Kay 1998; Hardy and Wells 2004; Gaczynska and Osmulski 2005; Goodey and Benkovic 2008).

Conventionally, allosteric coupling strengths between two processes have been demonstrated indirectly, through changes in activity, e.g. binding affinity (Kolb and Weber 1975; Kolb and Weber 1975; Cooper, McAlpine et al. 1994; Popovych, Sun et al. 2006; Muralidhara, Negi et al. 2007) or catalytic activity (Brennan, Christianson et al. 1995; Huang, Shiva et al. 2005; Gladwin and Kim-Shapiro 2008), that result from effector molecules binding to regulatory sites. However, there are limitations to such approaches. For example, they cannot be used in systems where there are no quantitatively assayable readouts. Coupling strength is also difficult to measure in situations where saturation with effector

molecules is not achievable. Additionally, allosteric coupling strength is most readily measured in terms of thermodynamics. That is, how does one equilibrium affect the population of states across another? However, equilibria can also be coupled purely (or at least largely) kinetically, with one equilibrium affecting the rates of transitions across another rather than the populations. Such kinetic coupling also plays an important role in certain cases of allosteric regulation (Nicholson, Munson et al. 1998; Boehr, Dyson et al. 2006; Boehr, McElheny et al. 2006; Goodey and Benkovic 2008). The indirect experimental approach is poorly suited to characterize kinetic coupling strengths in most cases. Finally, there would also be fundamental value in quantifying coupled equilibria by direct analysis of the equilibria themselves, rather than indirectly through the proxy of activity. In Chapter 3, I presented a strategy to directly parameterize four-state equilibria derived from two coupled dynamic processes. This strategy circumvents the requirement for ligand saturation and assayable readouts, and can enable quantification of both kinetic and thermodynamic coupling strengths.

In Chapter 2, I described two dynamic processes in the autoinhibited DH domain of the guanine nucleotide exchange factor, Vav1 (Li, Martins et al. 2008). One process is intrinsic to the DH domain, the two states of which are arbitrarily depicted using a square and a circle in Figure 4-1a. The other process involves the inhibitory helix associating with and dissociating from the DH domain (Fig. 4-1a). Intriguingly both thermodynamics and kinetics of the intrinsic process

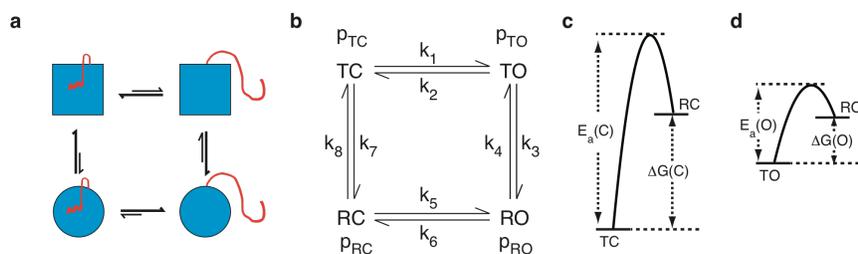


Figure 4-1. The four-state model derived from two processes and the hypothetical energy diagrams of one process. A pictorial (a) or schematic (b) model describes an intrinsic process coupled with an allosteric process. The square to circle equilibrium depicts the intrinsic process and the square and the circle are arbitrarily represented by T and R, respectively. The inhibitory helix close-open equilibrium is the allosteric process. The close and open state are represented by the C and O, respectively. Combination of the intrinsic process and the allosteric process gives rise to a circular four-state equilibrium. The fractional populations are p_{TC} , p_{TO} , p_{RO} and p_{RC} for TC, TO, RO and RC, respectively. The first order rate constants k_1 -8 are as indicated. (c) A hypothetical energy diagram of the intrinsic process when the helix is bound. $E_a(C)$ and $\Delta G(C)$ are the activation energy barrier of the transition from T state to R state and the free energy difference between the two states in the presence of the allosteric ligand. (d) A hypothetical energy diagram of the intrinsic process when the helix is dissociated and melted. $E_a(O)$ and $\Delta G(O)$ are the activation energy barrier of the transition from T state to R state and the free energy difference between the two states in the absence of the allosteric ligand.

change upon perturbations via phosphorylation, mutation or truncation towards the allosteric process, suggesting that the two dynamic processes are allosterically coupled (Li, Martins et al. 2008). To provide further insight into Vav1 function, we investigate the energetic and kinetic coupling strengths of the two processes in the autoinhibited Vav DH domain by direct parameterization of the four-state equilibrium using the strategy in Chapter 3. The energetic and kinetic coupling strengths of the two dynamic processes in AD are $1.0 \sim 1.5 \text{ kcal M}^{-1}$.

Results

The four-state model in the Vav1 DH domain

The autoinhibited Vav1 construct used in this work will be referred to as AD, as it contains the inhibitory helix from the Acidic region and the DH domain of the protein. A schematic representation of the four-state equilibrium in AD is shown in Figure 4-1b. The two states of the intrinsic process are referred as the T state (square in Fig. 4-1a) and the R state (circle in Fig. 4-1a) following the nomenclature in the MWC model (Monod, Changeux et al. 1963; Monod, Wyman et al. 1965). The helix open-close process is the allosteric process and referred to as the CO process reflecting the physical state of the inhibitory helix, i.e. C for closed and O for open (Fig. 4-1b). The combination of the TR process and the CO process gives rise to a circular four-state equilibrium as depicted in Figure 4-1b. The left and right vertical edges are referred as the closed edge and the open edge, respectively.

The free energy differences between the T state and the R state are $\Delta G(\text{TR})_C$ and $\Delta G(\text{TR})_O$ for the helix in the closed and open states, respectively (Figs. 4-1c and 4-1d). When the TR and CO equilibria are thermodynamically coupled, $\Delta G(\text{TR})_C$ is different from $\Delta G(\text{TR})_O$ due to different states of association of the inhibitory helix with the DH domain. The magnitude of this coupling, $\Delta\Delta G$, is defined as:

$$\Delta\Delta G = |\Delta G(\text{TR})_C - \Delta G(\text{TR})_O| = RT * \left| \ln\left(\frac{p_{TC}}{p_{RC}}\right) - \ln\left(\frac{p_{TO}}{p_{RO}}\right) \right| \quad (1)$$

The activation energies for the T to R transitions are defined as $E_a(\text{TR})_C$ and $E_a(\text{TR})_O$ for the helix in the closed and open state, respectively (Figs. 4-1c and 4-1d). $E_a(\text{TR})_C$ and $E_a(\text{TR})_O$ are directly related to the corresponding microscopic rate constants (k_7 and k_3 in Figure 4-1b) according to Arrhenius equation:

$$E_a = -RT \ln\left(\frac{k}{A}\right) \quad (2)$$

in which R is the ideal gas constant, T is absolute temperature, k is the corresponding microscopic rate constants and A is a scaling factor. Association or dissociation of the helix can also modulate the activation energy for the T to R transition differentially. The modulation is the origin of kinetic coupling, ΔE_a , which is defined as:

$$\Delta E_a = |E_a(\text{TR})_C - E_a(\text{TR})_O| = RT * \left| \ln\left(\frac{\frac{k7}{A}}{\frac{k3}{A}}\right) \right| = RT * \left| \ln\left(\frac{k7}{k3}\right) \right| \quad (3)$$

in which the scaling factors for $E_a(\text{TR})_C$ and $E_a(\text{TR})_O$ are assumed to be the same. According to the thermodynamic constraint, populations of the thermodynamic box are totally determined by its kinetic parameters. For example,

$$\frac{p_{TC}}{p_{RC}} = \frac{k8}{k7} \quad (4)$$

$$\frac{p_{TO}}{p_{RO}} = \frac{k4}{k3} \quad (5)$$

Incorporating Equations 4 and 5 into Equation 1, $\Delta\Delta G$ is converted into:

$$\Delta\Delta G = RT * \left| \ln\left(\frac{k8}{k7}\right) - \ln\left(\frac{k4}{k3}\right) \right| = RT * \left| \ln\left(\frac{k8}{k4}\right) - \ln\left(\frac{k7}{k3}\right) \right| \quad (6)$$

Therefore $\Delta\Delta G$ (Equation 6) and ΔE_a (Equation 3) are two intimately related and yet distinct quantities. $\Delta\Delta G$ can be zero while ΔE_a is not, *vice versa*.

Parameterization of the four-state equilibrium

According to the strategy in Chapter 3, single quantum (SQ) ^{13}C methyl CPMG data of AD and an open mutant, AD_{K208E} (Li, Martins et al. 2008), were acquired at 600 MHz and 800 MHz field strengths at 5 °C. The open edge of the four-state equilibrium of AD is fully characterized by globally fitting CPMG data

of 14 resonances of AD_{K208E} to a two-state model (Fig. 3-1a). Fitted kinetic/thermodynamic (k_{ex2} and p_{TO}/p_{RO} at 5 °C in Table 4-1) and chemical shift parameters were used as constraints in curve fitting of the CPMG data for AD. Therefore there remain three adjustable kinetic parameters, k_{ex1} , k_{ex3} , k_{ex4} . Because the sum of the four populations is unit, which introduces one more constraint into the four thermodynamic parameters, two thermodynamic parameters (p_{TC} and p_{RC}), remain to be determined. Since the chemical shift differences between states dictate relaxation dispersion, there are two adjustable chemical shifts for each resonance in AD upon the addition of one chemical shift difference from AD_{K208E}. During 4-state curve fitting of the AD data, a gradient descent search was carried out from finely divided grids of adjustable kinetic/thermodynamic parameters (See Methods section for details). One solution, i.e. a minimum on the χ^2 surface, was obtained through gradient descent search from each of the 240 grid points. Solutions obtained from different starting points can be potentially different, which is especially true for a rough χ^2 surface. In order to systematically determine the relationships between all solutions obtained, the logarithms of average normalized pair-wise distances (LAND) of the 240 solutions were calculated according to the following equation:

$$\text{LAND}(S1 \leftrightarrow S2) = \max(-1.5, \log_{10} \sqrt{\frac{\sum_{i=1}^n \frac{(S1(i) - S2(i))^2}{(|S1(i)| + |S2(i)|)^2}}{n}}) \quad (7)$$

in which $\text{LAND}(S1 \leftrightarrow S2)$ represents LAND of n corresponding elements of solution 1 and solution 2. LAND values below -1.5, including diagonal distances, were set to -1.5, which corresponds to an upper limit of 6.3% deviation per adjustable parameter for the two solutions in comparison. Plots of LAND for the 5 adjustable kinetic/thermodynamic parameters and 48 chemical shift parameters (from 24 resonances) are shown in Figures 4-2a and 4-2b, respectively. The corresponding χ^2 values are also shown (Fig. 4-2c). A majority of the solutions have comparable χ^2 . In fact, 184 out of the 240 solutions possess χ^2 between 164 and 170 (solution 9 to solution 192 in Fig. 4-2c). Computer simulation revealed that adjustable kinetic/thermodynamic parameters suffer less from inter-parameter correlation than chemical shifts (Fig. 3-3d-e in Chapter 3). Consistent with such observation, more than half of the solutions obtained for AD are close (LAND < -1.5) in terms of kinetic/thermodynamic parameters (Fig. 4-2a) and all of them have χ^2 below 170 (Fig. 4-2c). Chemical shifts, however, are more poorly determined than kinetic and thermodynamic parameters (Fig. 4-2b) due to sign swapping problem discussed in Chapter 3.

As noted in Chapter 3, it is inappropriate to use χ^2 as the sole criteria for global solution determination due to the high complexity of the four-state model. Rather, orthogonal information should be employed in conjunction with χ^2 to determine the global solution. Following the same logic as in Chapter 3, the

global solution of AD should recapitulate the experimental chemical shift differences between heteronuclear single quantum coherence spectra (HSQC) of AD and that of AD_{K208E} (CSD^{HSQC}). CSD^{HSQC} was first obtained by comparing HSQC spectra of AD and AD_{K208E} as in Chapter 2 (Li, Martins et al. 2008). Then chemical shift differences between AD and AD_{K208E} (CSD^{CPMG}) were calculated based on fitted parameters in all solutions of χ^2 less than 170. The slopes and coefficients of determination (r^2) of the linear correlations of CSD^{HSQC} with CSD^{CPMG} for all solutions were determined. To evaluate which solutions recapitulate CSD^{HSQC} the best, slopes versus r^2 for the solutions are plotted in Figure 4-2d. The solutions with higher r^2 tend to have slopes closer to unity, which is marked by the black horizontal dotted line in Figure 4-2d. There are 36 solutions of r^2 beyond 0.9, which is marked by the black vertical dotted line in Figure 4-2d and the slopes of them are between 0.95~1.1 (inset in Fig. 4-2d). Furthermore, there are 7 solutions possessing r^2 values larger than 0.92 (marked by the red vertical dotted line in Fig. 4-2d), all of which have slopes very close to unity (inset in Fig. 4-2d). The correlation plot giving the largest r^2 is shown in Figure 4-2e. The slope is 1.00 ± 0.04 , very close to unity, and r^2 is 0.935. Experimental CPMG curves agree very well with the ones calculated based on this solution as shown in Figure 4-3. The goodness of the fitting suggests that the circular four-state model is sufficient to represent the dynamic landscape of AD and that this solution is the global solution for this model. The populations of TC,

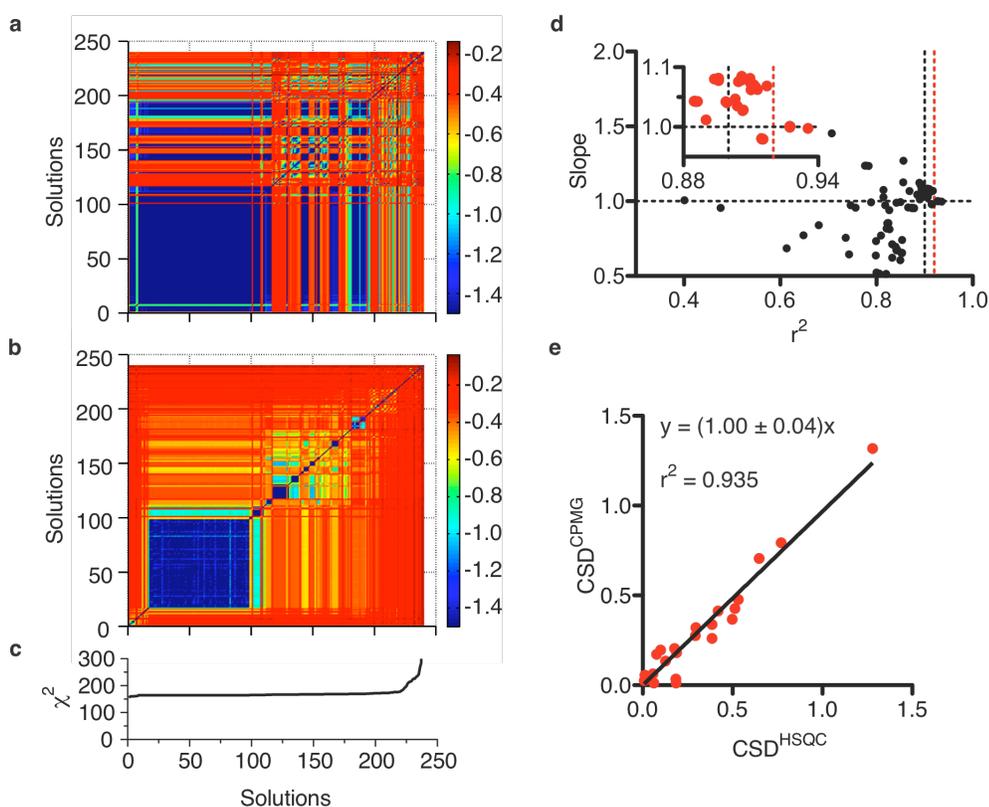


Figure 4-2. Solve the four-state equilibrium of AD. **(a)** Common logarithms of normalized pair wise distances of the kinetic/thermodynamic parameters (k_{ex1} , k_{ex3} , k_{ex4} , p_{TC} and p_{RC}). **(b)** Common logarithms of normalized pair wise distances of the chemical shifts. All distances below $10^{-1.5}$, including diagonal distances, are set to $10^{-1.5}$. The corresponding χ^2 are plotted in **(c)**. The ranking of the solutions on the x-axis of **(a-c)** and the y-axis of **(a-b)** is based on the ascending order of the corresponding χ^2 . The χ^2 of the last three solutions are larger than 300 and therefore not shown in **(c)**. **(d)** The slopes and the coefficients of determination (r^2) of the linear correlations of chemical shift difference of AD and K208E calculated from the solutions (CSD^{CPMG}) with chemical shift difference obtained from HSQC of AD and K208E (CSD^{HSQC}) are plotted against each other. The black and red dotted vertical line is marking 0.90 and 0.92, respectively, on x-axis and the dotted horizontal line is marking unit on y-axis. There are 36 and 7 data points with r^2 larger than 0.90 and 0.92, respectively (inset of **(d)**). Among the 7 points beyond 0.92, 6 are overlapping to give the point on the left in the axis scales. The solution with largest χ^2 is taken as the global solution although all 7 solutions of χ^2 larger than 0.92 possess identical kinetic/thermodynamic parameters. The CSD^{CPMG} versus CSD^{HSQC} plot of the global solution is shown in **(e)**. The solid line is the best fitted line with the assumption that the line goes through (0.0, 0.0).

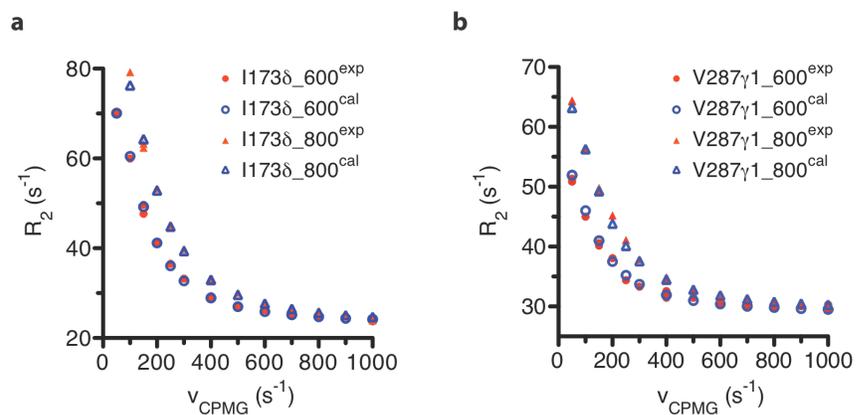


Figure 4-3. Representative CPMG relaxation dispersion curves. Experimental (red solid symbols) and best fitted (blue open symbols) 600 MHz (circle) and 800 MHz (triangle) relaxation dispersion curves of methyl I173δ (a) and V287γ1 (b) are shown.

TO, RO and RC are 0.84 ± 0.01 , 0.084 ± 0.013 , 0.029 ± 0.01 and 0.047 ± 0.01 , respectively (Table 4-1). Kinetic parameters k_{ex1} , k_{ex2} , k_{ex3} and k_{ex4} are 760 ± 50 , 1860 ± 70 , 600 ± 300 , $690 \pm 60 \text{ s}^{-1}$ (Table 4-1).

Robustness of the global solution

Next a Monte Carlo (MC) algorithm was used to investigate the robustness of the global solution toward random noise. A total of 1000 MC simulations were carried out. In each iteration, we added 4 % random noise to noise-free data calculated from the global solution. These data were then fit using a gradient descent search from the global solution. Histograms of $\ln(k_{ex1})$, $\ln(k_{ex3})$, $\ln(k_{ex4})$, p_{TC} , and p_{RC} are shown in Figures 4-4a to 4-4e. MC simulation results suggest that populations and k_{ex} s on substantially populated edges are well defined. Among the kinetic and thermodynamic parameters, k_{ex3} is the most poorly defined because its contributions to relaxation are marginal due to the low populations of RC and RO. Means and standard deviations of the fitted parameters from MC simulation are reported along with the global solution in Figure 4-4f. The two are statistically indistinguishable from each other (Fig. 4-4f). The fitted kinetic and thermodynamic parameters from MC simulation are also shown in Table 4-1.

Model Selection

Even though the formal description of the AD system involves 2 different equilibria and thus 4 states, one can't assume that the data are sufficient to

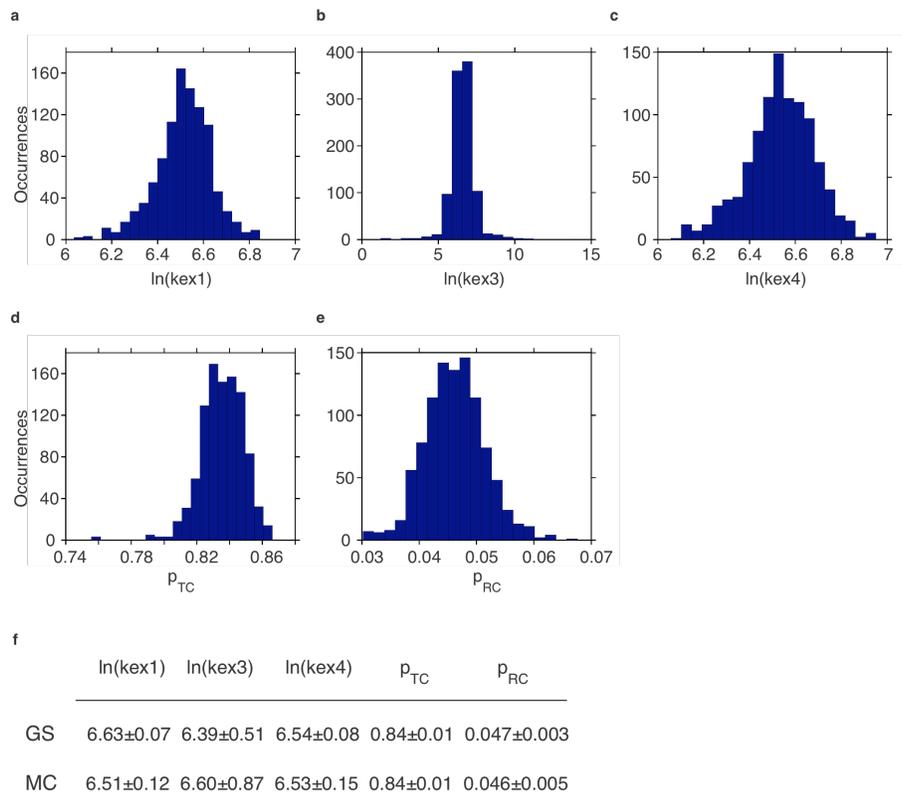


Figure 4-4. Monte Carlo simulation results. Histograms of $\ln(kex1)$ (a), $\ln(kex3)$ (b), $\ln(kex4)$ (c), p_{TC} (d), p_{RC} (e) from 1000 runs of MC simulation are shown. Comparison of the MC results with the global solution is shown in (f).

characterize the process at this level of detail. In addition, the fact that the circular four-state model (referred to as C4 hereafter, see Fig. 4-5a for details) is sufficient to account for the dynamic landscape of AD also does not justify the necessity to invoke such a complicated model instead of a simpler one. If either k_{ex1} or k_{ex3} is so small that it is out of the CPMG-sensitive regime, the model reduces to the first or the second linear 4-state model (L4_1 and L4_2 in Fig. 4-5a). If k_{ex4} is very fast, C4, L4_1 and L4_2 are further reduced to the circular 3-state model (C3), the first linear 3-state model (L3_1) and the second linear 3-state model (L3_2), respectively (Fig. 4-5b). A two-state model isn't considered here since initial analysis of the AD data clearly showed that this was insufficient to explain the data (Chapter 2 and Li, Martins et al. 2008). I used the F-test to determine whether C4 is indeed the most appropriate model for AD. I first carried out gradient descent search of the AD data at 5 °C from multiple initial points assuming the models C3, L3_1, L3_2, L4_1 or L4_2. As observed for the C4 model, each of these other models also gave multiple solutions of comparable χ^2 from different starting points on the grid (Fig. 4-2c for C4, data not shown for other models). I carried out the F-test for the lowest χ^2 value and for the median of the lowest hundred χ^2 values for all six models (second column in Fig. 4-5c). There are 822 independent experimental measurements in the AD dataset. The number of adjustable parameters is 100, 99, 99, 74, 73 and 73 for C4, L4_1, L4_2, C3, L3_1 and L3_2, respectively. The denominator degree of freedom is 721 (=

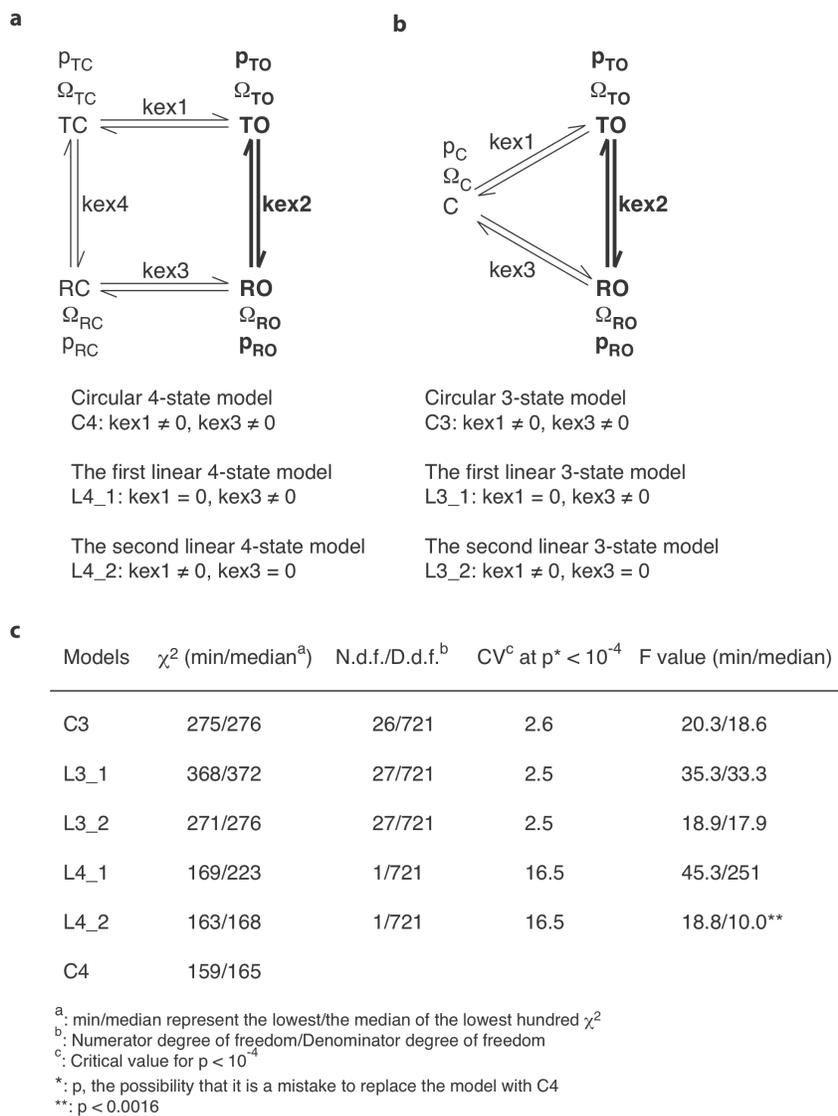


Figure 4-5. F-test distinguishes the circular 4-state model from other simpler models for AD. **(a)** The circular and two linear 4-state models. The circular 4-state model in Figures 1a-b is reproduced here and referred as C4. Kinetic parameters k_{ex1} , k_{ex2} , k_{ex3} and k_{ex4} are defined as k_1+k_2 , k_3+k_4 , k_5+k_6 and k_7+k_8 , respectively. In addition, Ω_{TC} , Ω_{TO} , Ω_{RO} and Ω_{RC} are chemical shifts of states TC, TO, RO and RC, respectively. If k_{ex1} or k_{ex3} is very small, the model can be simplified as the first or the second linear 4-state, i.e. L4_1 or L4_2. **(b)** The circular and two linear 3-state models. When k_{ex4} is very fast, TC and RC can be treated as one species as far as NMR properties is concerned and C4, L4_1 and L4_2 are further simplified as a circular 3-state (C3), the first linear 3-state model (L3_1) and the second linear 3-state model (L3_2), respectively. In these models, states TC and RC collapse into one state C and p_C and Ω_C are the fractional population and the chemical shift of state C, respectively. In both **(a-b)**, the open edge is highlighted in bold to reflect the fact that this edge is to be characterized independently using an open construct and parameters on this edge are therefore constrained in curve fitting of AD data. **(c)** F-test results to distinguish C4 from others. Due to the complexity of the models, there is more than one solutions of comparable χ^2 obtained from multiple starting initial points. Therefore the minimum χ^2 of all and the median of the lowest hundred χ^2 of each model are compared (second column). Given the numerator and denominator degree of freedoms (third column), the critical value for $p < 10^{-4}$ (fourth column) and the corresponding F values were calculated (last column).

822 - (100 + 1)) and the numerator degrees of freedom are 26, 27, 27, 1 and 1 for C3, L3_1, L3_2, L4_1 and L4_2, respectively (the third column in Fig. 4-5c). For the tests based on the lowest χ^2 , the critical values for $p < 10^{-4}$ are 2.6, 2.5, 2.5, 16.5 and 16.5 for C3, L3_1, L3_2, L4_1 and L4_2, respectively. The F values for C3, L3_1, L3_2 and L4_1 are much larger than the corresponding critical values (Fig. 4-5c). Even the F value based on lowest χ^2 for L4_2 is also higher than the critical value of 16.5. The p value is less than 0.0016 for L4_2 using the median of the lowest hundred χ^2 . This analysis suggests that C4 is indeed the most appropriate model for AD at 5 °C. Below I assume C4 to be the appropriate model for AD at higher temperatures and for AD_{K208A}.

CPMG curve fitting recapitulates open populations measured independently

Using chemical shift information obtained from AD data at 5 °C, together with kinetic and thermodynamic information on AD_{K208E} derived from CPMG data acquired at 10 and 15 °C, I also analyzed data on AD acquired at 10 and 15 °C separately. For each temperature, there are three kinetic adjustable parameters, kex1, kex3, and kex4, and two thermodynamic parameters, p_{TC} and p_{RC}, as for 5 °C data analysis. Owing to complete constraints on chemical shifts, the kinetic and thermodynamic parameters were precisely determined except for kex3, which makes an intrinsically low contribution to relaxation because of the low populations of the R edge (Table 4-1).

In addition to AD_{K208E}, in which helix-DH interaction is completely disrupted, a series of mutants that differentially populate the closed and open conformations were also reported in a recent study (Chapter 2 and Li, Martins et al. 2008). One of these, AD_{K208A}, was estimated to have an 0.77:0.23 ratio of closed:open (TC+RC:TO+RO), based on chemical shifts and Y174 phosphorylation kinetic data. Relaxation dispersion data were recorded on perdeuterated AD_{K208A} at 600 and 800 MHz, at 5 and 10 °C. Curve fitting of AD_{K208A} data at 5 and 10 °C were also carried out separately using kinetic and thermodynamic information of the closed and open edge of AD at the same temperatures. For each temperature, there are two kinetic adjustable parameters, k_{ex1} and k_{ex3} , and one thermodynamic parameter, p_{TC} . Dispersion data for two more resonances, V287 γ 2 and L325 δ 2, become available for AD_{K208A}, which overlap with other resonances in AD spectra. Chemical shifts of the remaining resonances for AD_{K208A} were constrained by information from AD at 5 °C. So there are totally 4 adjustable chemical shifts. Natural logarithms of k_{ex1} are 7.25 ± 0.04 and 7.83 ± 0.05 for 5 and 10 °C, respectively; p_{TC} is 0.107 ± 0.002 and 0.126 ± 0.006 for 5 and 10 °C, respectively. The large number of constraints allowed k_{ex1} and p_{TC} to be well determined. Natural logarithms of k_{ex3} is 8.2 ± 0.4 for 5 °C. It is very poorly determined for 10 °C. The poor determination of k_{ex3} is due to the low populations of the R edge and therefore an intrinsically low contribution to relaxation.

Previous work gives an independent assessment of the total open and closed populations (Chapter 2 and Li, Martins et al. 2008). Comparison of these independently determined open populations with the values determined from relaxation provides an independent assessment of the correctness of the 4-state analysis here. Therefore I experimentally determined the open population of AD, p_{open} , at 5, 10 and 15 °C and of AD_{K208A} at 5 and 10 °C as in a recent study (Li, Martins et al. 2008). The total open populations were also obtained from fitted parameters by summing up p_{TO} and p_{RO} of the global solutions. Figure 4-6 shows the plot of the fitted open populations (as $p_{\text{TO}}+p_{\text{RO}}$) against the independently determined open populations, p_{open} . A linear fit with the slope close to unity and $r^2 > 0.99$ was obtained. The high degree of agreement between p_{open} and $p_{\text{TO}}+p_{\text{RO}}$ further indicates that the strategy established in Chapter 3 can be successfully solve four-state equilibria in AD and AD_{K208A}.

Energetic and kinetic coupling strengths

The population ratios of the T state over the R state are 2.9, 3.1 and 2.4 at 5, 10, and 15 °C, respectively, in the helix-dissociated (open) state (Table 4-1, right arm in Fig. 4-1b). When the helix is bound to the DH domain the ratios increase 6.1-, 6.6- and 6.4-fold at 5, 10, and 15 °C, respectively (left arm in Fig 4-1b). The population increases correspond to energetic coupling strengths ($\Delta\Delta G$) between 1.0 and 1.1 kcal M⁻¹ (Table 4-1) between the TR and CO equilibria.

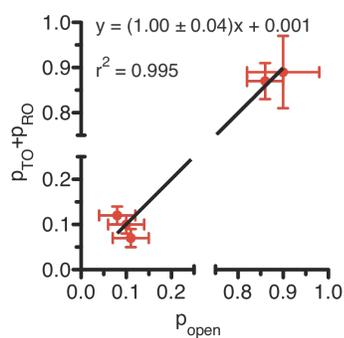


Figure 4-6. Correlation between experimentally determined open populations with fitted ones from CPMG. The experimentally determined open populations were obtained from HSQC of AD and AD_{K208A}. The fitted open populations were obtained by summing up p_{TO} and p_{RO} . The slope of the line is 1.00 ± 0.04 and the coefficient of determination, r^2 , is 0.995.

Table 4-1 Fitted parameters and kinetic and thermodynamic coupling strengths of AD

	5 °C	10 °C	15 °C
kex1	6.63 ± 0.07	7.43 ± 0.03	7.77 ± 0.07
kex2	7.53 ± 0.04	8.15 ± 0.04	8.82 ± 0.12
kex3	6.39 ± 0.51	ill defined	ill defined
kex4	6.54 ± 0.08	7.62 ± 0.04	8.59 ± 0.02
p _{TC}	0.84 ± 0.01	0.86 ± 0.01	0.88 ± 0.01
p _{TO}	0.084 ± 0.013	0.074 ± 0.014	0.045 ± 0.015
p _{RO}	0.029 ± 0.01	0.024 ± 0.01	0.018 ± 0.01
p _{RC}	0.047 ± 0.01	0.042 ± 0.01	0.057 ± 0.01
ΔΔG (folds)	6.1 ± 2.7	6.6 ± 3.4	6.4 ± 4.2
ΔΔG (kcal M ⁻¹)	1.00 ± 0.24	1.06 ± 0.29	1.06 ± 0.27
ΔE _a (folds)	12.9 ± 6.0	8.9 ± 4.8	6.0 ± 3.1
ΔE _a (kcal M ⁻¹)	1.41 ± 0.26	1.23 ± 0.31	1.03 ± 0.40

With the helix associated, the transition rates from the T state to the R state are 36, 95 and 327 s⁻¹ at 5, 10, and 15 °C, respectively. Dissociation of the inhibitory helix causes 12.9-, 8.9- and 6.0-fold increase in the transition rates, which translate to kinetic coupling strengths (ΔE_a) 1.4, 1.2 and 1.0 kcal M⁻¹ at 5, 10, and 15 °C, respectively (Table 4-1).

Discussion

Energetically and kinetically coupled equilibria form the basis of allosteric regulation throughout biology. Allostery has been extensively studied for more than half a century. A well-known example is the binding of initial oxygen molecules to hemoglobin facilitates the binding of subsequent molecules, i.e. these binding events are energetically coupled (Monod, Changeux et al. 1963). Recently, Wright and coworkers elegantly demonstrated the energetic and kinetic coupling in the enzymatic cycling of *E. coli* dihydrofolate reductase (DHFR) (Boehr, McElheny et al. 2006). Each intermediate in the five-step catalytic cycle of DHFR samples excited states resembling the ground state of preceding and following intermediates. The following-up events paddle the cycle around unidirectionally through energetic coupling to the pre-equilibrium in the preceding intermediates. In the last step the enzyme/product binary complex is extremely slow in product release. Cofactor binding substantially accelerates the rates of sampling a higher-energy conformation, which is fast in product release, suggesting the product release process is kinetically coupled to cofactor binding.

Multiple processes are prevalent in multi-domain and single-domain proteins (Eisenmesser, Bosco et al. 2002; Eisenmesser, Millet et al. 2005) and these processes are often allosterically coupled (Eisenmesser, Millet et al. 2005) (Yu, Martins et al. submitted, Martins et al. manuscript in preparation). Quantification of energetic coupling and kinetic coupling strengths is much less accessible in these proteins than in DHFR since deconvolution of individual processes is nontrivial. To address situations like these, a strategy was developed to directly characterize kinetic and energetic coupling strengths between two processes based on CPMG measurements (Chapter 3). In the current Chapter, this technique is used to characterize the energetic and kinetic coupling strengths of two previously observed dynamic processes in AD (Li, Martins et al. 2008). The energetic and kinetic coupling are $1.0 \sim 1.1 \text{ kcal M}^{-1}$ and $1.0 \sim 1.5 \text{ kcal M}^{-1}$, respectively. In a construct contain the whole regulatory apparatus of Vav, the helix close-open process is further subject to thermodynamic modulation of other domain-domain interactions that are absent in AD (Chapter 2 and Yu, Martins et al. submitted). Recent studies based on a variety of indirect methodologies reveal that the coupling strength between the helix process and other domain-domain interactions is about $1.4 \text{ kcal mol}^{-1}$ (Chapter 2 and Yu, Martins et al. submitted). These data suggest energetic and kinetic coupling strength in allosteric proteins are probably a few kcal mol^{-1} . A welcome consequence of the coupling strengths being relatively low is that it is right in the CPMG amenable regime (Palmer,

Kroenke et al. 2001). Our work represents, to the best of our knowledge, the first quantitative analysis of dynamics of this complexity, and will provide a route to solving related problems in the future. It also represents the first direct quantitative measurement of the degree of thermodynamic coupling between two dynamic processes in an allosteric system. Finally, the work provides a detailed physical understanding of Vav, an important human proto-oncoprotein.

Materials and methods

NMR spectroscopy. U- ^{15}N , ^2H], Ile- $^{13}\text{C}^\delta\text{H}_3$ -, Leu- $^{13}\text{CH}_3$, $^{12}\text{CD}_3$], Val- $^{13}\text{CH}_3$, $^{12}\text{CD}_3$]-labeled proteins were expressed and purified as previous reported (Li, Martins et al. 2008). NMR samples contained c.a. 1.0 mM freshly purified protein in 25 mM phosphate buffer (pH = 7.2) with 50 mM NaCl, 5 mM DTT, 1 mM EDTA, 0.1 % (w/v) NaN_3 . Single quantum ^{13}C methyl Carr-Purcell-Meiboom-Gill (CPMG) relaxation dispersion experiments were acquired at 600, 800 and 900 MHz Varian Inova spectrometers at 5, 10, and 15 °C using a pulse sequence kindly provided prior to its publication by Dr. Lewis E. Kay (Lundstrom, Vallurupalli et al. 2007). All CPMG relaxation dispersion experiments used a constant relaxation period of 40 milliseconds. For each CPMG dispersion curve, 18 to 20 ^1H - ^{13}C methyl HSQC were recorded with ν_{CPMG} frequencies ranging from 50 to 1000 Hz, among which there were at least three pairs of duplicates. Effective transverse relaxation rates, $R_{2,\text{eff}}$ were obtained from peak intensities obtained on the methyl HSQC via:

$$R_{2,\text{eff}}(\nu_{\text{CPMG}}) = -\frac{1}{T} \ln \frac{I(\nu_{\text{CPMG}})}{I_0} \quad (8)$$

where T is the length of the constant time period, $I(\nu_{\text{CPMG}})$ is the intensity recorded with a given ν_{CPMG} value, and I_0 is the intensity in a common reference spectrum recorded without the constant relaxation time interval. The average of the standard deviations of the duplicate points was assumed the uncertainty of each point of the CPMG profile. If the calculated uncertainties were less than 4% of the corresponding R_2 , a value of 4% was employed. The experimental open populations (p_0) for AD and AD_{K208A} were calculated based on an ideal open state mimic mutant, AD_{K208E} and an ideal closed state mimic mutant, AD_{K169E207K}, as described previously (Li, Martins et al. 2008).

Curve fitting. The curve fitting for the 2- and 4-state models was carried out as in Chapter 3. The evolution matrices for the 3-state models are the same as in Tolkatchev et al. (Tolkatchev, Xu et al. 2003).

CPMG data at 600MHz and 800MHz of 14 resonances of AD_{K208E} at 5 °C were fitted globally to the two-state model (Fig. 3-1b). Data of the same 14 resonances of AD_{K208E} from 600MHz and 800MHz at 10 °C were fitted globally to the two-state model with the fitted chemical shift changes at 5 °C used as constraints. Both 600MHz and 900MHz data of the 14 resonances and 600MHz data of L243δ1 of AD_{K208E} at 15 °C were fitted globally to the two-state model

with the fitted chemical shift changes of the 14 resonances at 5 °C used as constraints. The fitted chemical shift changes of the 15 resonances were used as constraints for the open edge of the three-state models or the four-state models (Figs. 4-5a and 4-5b) in the process of curve fitting of CPMG data of AD and AD_{K208A}, i.e. $\Omega_{TO}=0.0$ and Ω_{RO} takes the value of the fitted chemical shift changes. It is worth noting that the signs of chemical shifts were not considered due to its symmetry in determining single quantum relaxation. The fitted kexs and populations of K208E were also used as constraints for the open edge of the three- and four-state models in the process of curve fitting. That is to say, p_T/p_R and kex of K208E were used as constraints for p_{TO}/p_{RO} and kex2 of the three- and four-state models at the same temperatures.

In the process of three-state curve fitting, kex2 of AD was assumed the fitted kex of K208E. The initial values of kex1 and kex3 were set to the same and nonzero for circular 3-state model (C3, Fig. 4-5b) or kex1 and kex3 were set to zero for the first linear 3-state model (L3_1, Fig. 4-5b) and the second linear 3-state model (L3_2, Fig. 4-5b), respectively. Natural logarithms of nonzero kex1 and/or kex3 ranged from 4.0 to 8.0 with the step size of 0.2. Since AD is largely closed, p_C scanned from 0.56 to 0.76 with the step size 0.04. The last $R_{2,eff}$ value of each curve was used as the initial values of the corresponding R_2^0 . For the intrinsic process resonances, Ω_{TO} and Ω_{RO} were fixed to zero and initial Ω_C were

set to 0.4 ppm except that initial Ω_C of V328 γ 2 was set to 1.4 ppm given its severity in line broadening and high amplitude of relaxation dispersion. For the allosteric process resonances, the fitted chemical shifts of each resonance ($\Omega_T = 0$ and Ω_R) in K208E were used as constraints for Ω_{TO} ($= 0$) and Ω_{RO} in AD and initial Ω_C was set equal to 0.0 ppm.

In the process of four-state curve fitting of AD at 5 °C, k_{ex2} of AD protein was also assumed the fitted k_{ex} of K208E at 5 °C. The initial values of k_{ex1} , k_{ex3} and k_{ex4} were set to the same for the circular 4-state model (C4) or $k_{ex1} = 0$ and $k_{ex3} = k_{ex4} \neq 0$ for the first linear 4-state model (L4_1) or $k_{ex1} = k_{ex4} \neq 0$ and $k_{ex3} = 0$ for the second linear 4-state model (L4_2) (Fig. 4-5a). Natural logarithm of nonzero k_{ex1} , k_{ex3} and k_{ex4} ranged from 5.0 to 8.0 with the step size of 0.2. Since it is consistent with the evidence that p_{TC} is the major species on the closed edge, initial p_{TC} was set from 0.56 to 0.76 with the step size 0.04 and initial p_{RC} was uniquely defined by the equation: $p_{RC} = 0.8 - p_{TC}$. The last $R_{2,eff}$ value of each curve was used as the initial values of the corresponding R_2^0 . For the allosteric process resonances, Ω_{TO} and Ω_{RO} were fixed to zero and initial Ω_{TC} and Ω_{RC} were set to 0.4 or 0.6 ppm. For the intrinsic process resonances, the fitted chemical shifts of each resonance ($\Omega_T = 0$ and Ω_R) in K208E were used as constraints for Ω_{TO} ($= 0$) and Ω_{RO} in AD and initial Ω_{TC} and

Ω_{RC} were set equal to Ω_{TO} and Ω_{RO} , respectively. The normalized pair wise distances were calculated as in Chapter 3.

CPMG data of AD at 10 °C or 15 °C were globally fitted to C4 the same way as of 5 °C data except that the chemical shifts of AD at 5 °C were used as constraints. The initial values of $kex1$ and $kex3$ were set the same, the natural logarithm of which ranged from 5.0 to 8.8 with the step size 0.2. The initial values of p_{TC} scanned from 0.51 to 0.89 with the step size 0.02. CPMG data of K208A at 5 °C or 10 °C were globally fitted to C4 with the chemical shifts of AD at 5 °C as constraints. In addition, $kex2$ and $kex4$ of K208A were set as $kex2$ and $kex4$ of AD at the same temperatures and p_{TO}/p_{RO} and p_{TC}/p_{RC} of K208A were fixed to those of AD. The initial values of $kex1$ and $kex3$ were set the same, the natural logarithm of which ranged from 5.0 to 8.8 with the step size 0.2. The initial values of p_{TC} scanned from 0.01 to 0.21 with the step size 0.02.

For any resonance i , the chemical shift perturbation obtained from HSQC (CSD^{HSQC}) was calculated based the equation below:

$$CSD^{HSQC} = | \Omega_i(AD) - \Omega_i(K208E) | \quad (9)$$

in which $\Omega_i(AD)$ and $\Omega_i(K208E)$ are the ^{13}C chemical shift of resonance i in ^{13}C - 1H HSQC spectra of AD and K208E, respectively. For the same resonances, the CSD obtained from CPMG (CSD^{CPMG}) was calculated as in Chapter 3

F-test for model selection. F-test was used to distinguish the circular 4-state model (C4) from other simpler models for AD at 5 °C (Fig. 4-5). F values of comparing a simpler model with C4 was calculated according to the following formula:

$$F = \frac{\frac{\chi_{\text{other}}^2 - \chi_{\text{C4}}^2}{p2 - p1}}{\frac{\chi_{\text{C4}}^2}{(n - (p2 + 1))}} \quad (10)$$

in which n is the total number of data points; χ_{other}^2 and χ_{C4}^2 are the normalized sum square residuals of all data points for any other simpler model and C4, respectively; p1 and p2 are number of adjustable parameters for the simpler model and C4, respectively.

MC simulations. In order to assess robustness of fitted parameters, a Monte Carlo algorithm was employed as in Chapter 3.

Chapter 5 Concluding remarks

Proteins are intrinsically dynamic entities that transition among numerous conformational states. This feature of proteins has been explored extensively throughout evolution such that many conformations important for stability, regulation, and activity are preserved and modulated (Palmer, Kroenke et al. 2001; Kern and Zuiderweg 2003; Kay 2005). For a protein involving in complex formation, its static apo-conformation is often incompatible with ligand binding for reasons like inaccessibility of binding pocket (Eriksson, Baase et al. 1992), etc. For a protein subject to post-translational modifications, its static apo-conformation is often inaccessible to modifying enzymes (Aghazadeh, Lowry et al. 2000). The key in solving this conundrum lies in the fact that a ligand binding-competent conformation or a modifying enzyme-accessible formation is significantly populated amongst the dynamic ensemble of the proteins (Mulder, Mittermaier et al. 2001; Mulder, Skrynnikov et al. 2001; Skrynnikov, Mulder et al. 2001; Volkman, Lipson et al. 2001; Li, Martins et al. 2008). The existence of pre-equilibria between binding-incompatible or modification-inaccessible ground state conformations and binding-compatible or modification-accessible excited state conformations is certainly an operational mode in, but not limited to, multi-domain signaling proteins like Vav1. Solution NMR structure of AD (Aghazadeh, Lowry et al. 2000) and X-ray crystal structure of CADPZ (Yu, Martins et al.

submitted) demonstrate that the critical tyrosine residual Tyr174 is sandwiched between the inhibitory helix and the DH domain and it is inaccessible to tyrosine kinase. In AD, the inhibitory helix transiently (10%) dissociates from the DH domain and melts into an extended conformation, which is compatible with substrate requirement of tyrosine kinases. The population of the high-energy extended conformation quantitatively controls basal activity of AD and its full activation rate via tyrosine kinase (Li, Martins et al. 2008).

Intriguingly two truncated forms of Vav1, in which the inhibitory helix-DH interaction remains intact, exhibit transforming activity in mammalian cells (Bustelo 2000), suggesting that the abovementioned inhibitory equilibrium is insufficient to ensure appropriate regulation of Vav required *in vivo*. Other domain-domain contacts in full length Vav are thermodynamically coupled to the core inhibitory interactions and further bias the inhibitory equilibrium c.a. 10-fold more towards the inactive and kinase-inaccessible state (Fig. 2-13 and Yu, Martins et al. submitted). Many other multi-domain signaling molecules share the same inhibitory architecture with Vav, i.e. multiple weakly suppressing interactions co-exist and collectively provide strong ultimate suppression (Table 5-1). The presence of multiple weak interactions rather than a single strong interaction offers the molecules with the capability of rapid activation via integration of multiple signals (Yu, Martins et al. submitted). That could be one of the reasons why molecules of such architecture are so prevalent in signal

transduction pathways. Nevertheless, this corporative construction is another feature in many multi-signaling proteins.

The relationship of the modulatory and the core interactions in Vav1 is hierarchic, i.e. the modulatory interaction enhances suppression through coupling to the core interaction but not vice versa (Yu, Martins et al. submitted). The essence for the hierarchic architecture is not how strong the modulatory interaction is *per se*, but how much allosteric coupling strength the modulatory interactions exert on the core interactions. Deciphering the allosteric coupling strengths is pivotal in fully understanding the role of allostery in signaling molecules. Regulation through allosteric coupling is yet another operational mode in signaling molecules. Evolution has extensively explored the three operational modes: pre-equilibrium, collaboration of multiple weak interaction, and energetic coupling, among others, and offered multi-domain signaling molecules the capacity to fulfill their sophisticated and yet routine functions.

Table 5-1

Multiple weakly biasing interactions cooperatively provide strong suppression

Molecules	Interactions	References
FAK*	Δ FERM domain: 5-fold; Phosphorylated: 20-fold	(Lietha, Cai et al. 2007)
Hck	SH2 ligand: 2.5-fold; SH2/SH3 ligands: 6-fold	

(Moarefi, LaFevre-Bernt et al. 1997)

Abl Mutated SH3 binding: 7-fold; abrogate myristoylation: 7.5-fold;
both: 14-fold

(Hantschel, Nagar et al. 2003)

SH-PTP2 Mono-Phospho (SH2 ligand): 9-16-fold; Di-Phospho: 37-fold

(Pluskey, Wandless et al. 1995)

Tim Mutate SH3 binding: 3-fold; without inhibitory helix: 50-fold;
both: 75-fold

(Yohe, Rossman et al. 2008)

Vav1** Modulatory interaction: 10-fold; core interaction: 10-fold;
both: ~100-fold

(Yu, Martins et al. submitted)

*: Interactions for all molecules except Vav1 are activating interaction.

** : Interactions for Vav1 are inhibitory interactions.

In addition to identify potential operational mode in multi-domain signaling molecules, two novel strategies are also established in this thesis work to characterize allosteric coupling strengths in Vav. The first strategy is a conventional and indirect method, which is established to characterize the coupling strength between the modulatory interaction and the core interaction in CADPZ. The core interaction, i.e. the inhibitory helix binding and unbinding

with the DH domain (Aghazadeh, Lowry et al. 2000), controls the basal activity of AD (Fig. 2-11). In wild type AD, the core interaction possesses an open population of c.a. 10% and its GEF activity is about 10% of DH (Li, Martins et al. 2008). The modulatory interactions in CADPZ (Fig. 2-11) bias the core interaction further towards the inhibited form. The open population in CADPZ and its basal GEF activity are too low to be accurately measured. Mutations destabilizing the core interaction cause larger open population both in AD and in CADPZ. Importantly the same mutation causes less opening in CADPZ than in AD presumably due to the presence of the modulatory interactions. NMR and biochemical results from mutant AD and CADPZ allow accurate determination of the coupling strength between the modulatory interaction and the core interaction, which is about 10-fold (Table 5-1) (Yu, Martins et al. submitted). Such a coupling strength is consistent with the second operational mode: co-existence of multiple weak interactions. Since this strategy is readily applicable to other multi-domain systems, similar studies will certainly follow up to allow further assessment of generality of this mode. In the process of investigating the dynamic landscape of AD, two dynamic processes were discovered. Interestingly the two processes are thermodynamically and kinetically coupled, i.e. kinetics and thermodynamics of one process are perturbed upon modulations on the other process. A four-state equilibrium model is required to account for the dynamic landscape composed by the two processes. A second strategy is to characterize

thermodynamic and kinetic coupling strengths in AD by direct parameterization of the four-state equilibrium using NMR CPMG measurement. The coupling strengths are also in the weak category: $1.0 \sim 1.5 \text{ kcal M}^{-1}$. These two strategies are complementary with each other and collectively pave a new avenue for characterizing dynamic landscape of complex signaling molecules.

References:

- Abdul-Manan, N., B. Aghazadeh, et al. (1999). "Structure of Cdc42 in complex with the GTPase-binding domain of the 'Wiskott-Aldrich syndrome' protein." *Nature* **399**(6734): 379-83.
- Abe, K., I. P. Whitehead, et al. (1999). "Involvement of NH(2)-terminal sequences in the negative regulation of Vav signaling and transforming activity." *J Biol Chem* **274**(43): 30410-8.
- Abe, K., I. P. Whitehead, et al. (1999). "Involvement of NH(2)-terminal sequences in the negative regulation of Vav signaling and transforming activity." *J. Biol. Chem.* **274**: 30410-30418.
- Adams, J. M., H. Houston, et al. (1992). "The hematopoietically expressed vav proto-oncogene shares homology with the dbl GDP-GTP exchange factor, the bcr gene and a yeast gene (CDC24) involved in cytoskeletal organization." *Oncogene* **7**(4): 611-8.
- Aghazadeh, B., W. E. Lowry, et al. (2000). "Structural basis for relief of autoinhibition of the Dbl homology domain of proto-oncogene Vav by tyrosine phosphorylation." *Cell* **102**(5): 625-33.
- Akke, M. (2002). "NMR methods for characterizing microsecond to millisecond dynamics in recognition and catalysis." *Curr Opin Struct Biol* **12**(5): 642-7.
- Alexandropoulos, K. and D. Baltimore (1996). "Coordinate activation of c-Src by SH3- and SH2-binding sites on a novel p130Cas-related protein, Sin." *Genes Dev* **10**(11): 1341-55.
- Amarasinghe, G. K. and M. K. Rosen (2005). "Acidic region tyrosines provide access points for allosteric activation of the autoinhibited vav1 dbl homology domain." *Biochemistry* **44**(46): 15257-68.
- Arimura, N. and K. Kaibuchi (2005). "Key regulators in neuronal polarity." *Neuron* **48**(6): 881-4.
- Bachmann, M. F., L. Nitschke, et al. (1999). "The guanine-nucleotide exchange factor Vav is a crucial regulator of B cell receptor activation and B cell responses to nonrepetitive antigens." *J Immunol* **163**(1): 137-42.
- Bax, A. and A. Grishaev (2005). "Weak alignment NMR: a hawk-eyed view of biomolecular structure." *Curr Opin Struct Biol* **15**(5): 563-70.
- Beach, H., R. Cole, et al. (2005). "Conservation of mus-ms enzyme motions in the apo- and substrate-mimicked state." *J Am Chem Soc* **127**(25): 9167-76.
- Billadeau, D. D. (2002). "Cell growth and metastasis in pancreatic cancer: is Vav the Rho'd to activation?" *Int J Gastrointest Cancer* **31**(1-3): 5-13.
- Billadeau, D. D., J. L. Upshaw, et al. (2003). "NKG2D-DAP10 triggers human NK cell-mediated killing via a Syk-independent regulatory pathway." *Nat Immunol* **4**(6): 557-64.

- Blanchet, F., A. Cardona, et al. (2005). "CD28 costimulatory signal induces protein arginine methylation in T cells." *J Exp Med* **202**(3): 371-7.
- Boehr, D. D., H. J. Dyson, et al. (2006). "An NMR perspective on enzyme dynamics." *Chem Rev* **106**(8): 3055-79.
- Boehr, D. D., D. McElheny, et al. (2006). "The dynamic energy landscape of dihydrofolate reductase catalysis." *Science* **313**(5793): 1638-42.
- Boehr, D. D. and P. E. Wright (2008). "Biochemistry. How do proteins interact?" *Science* **320**(5882): 1429-30.
- Booden, M. A., S. L. Campbell, et al. (2002). "Critical but distinct roles for the pleckstrin homology and cysteine-rich domains as positive modulators of Vav2 signaling and transformation." *Mol Cell Biol* **22**(8): 2487-97.
- Brennan, C. A., K. Christianson, et al. (1995). "A molecular sensor system based on genetically engineered alkaline phosphatase." *Proc Natl Acad Sci U S A* **92**(13): 5783-7.
- Briggs, S. D. and T. E. Smithgall (1999). "SH2-kinase linker mutations release Hck tyrosine kinase and transforming activities in Rat-2 fibroblasts." *J Biol Chem* **274**(37): 26579-83.
- Burbulys, D., K. A. Trach, et al. (1991). "Initiation of sporulation in *B. subtilis* is controlled by a multicomponent phosphorelay." *Cell* **64**(3): 545-52.
- Burridge, K. and K. Wennerberg (2004). "Rho and Rac take center stage." *Cell* **116**(2): 167-79.
- Bustelo, X. R. (1996). "The VAV family of signal transduction molecules." *Crit. Rev. Oncog.* **7**(1-2): 65-88.
- Bustelo, X. R. (2000). "Regulatory and signaling properties of the Vav family." *Mol Cell Biol* **20**(5): 1461-77.
- Bustelo, X. R. (2001). "Vav proteins, adaptors and cell signaling." *Oncogene* **20**(44): 6372-81.
- Bustelo, X. R. and M. Barbacid (1992). "Tyrosine phosphorylation of the vav proto-oncogene product in activated B cells." *Science* **256**: 1196-1199.
- Bustelo, X. R., J. A. Ledbetter, et al. (1992). "Product of vav proto-oncogene defines a new class of tyrosine protein kinase substrates." *Nature* **356**(6364): 68-71.
- Cen, H., A. G. Papageorge, et al. (1992). "Isolation of multiple mouse cDNAs with coding homology to *Saccharomyces cerevisiae* CDC25: identification of a region related to Bcr, Vav, Dbl and CDC24." *EMBO J* **11**(11): 4007-15.
- Cheng, H. C., B. M. Skehan, et al. (2008). "Structural mechanism of WASP activation by the enterohaemorrhagic *E. coli* effector EspF(U)." *Nature* **454**(7207): 1009-13.
- Cole, R. and J. P. Loria (2002). "Evidence for flexibility in the function of ribonuclease A." *Biochemistry* **41**(19): 6072-81.

- Cooper, A., A. McAlpine, et al. (1994). "Calorimetric studies of the energetics of protein-DNA interactions in the E. coli methionine repressor (MetJ) system." *FEBS Lett* **348**(1): 41-5.
- Coppola, J., S. Bryant, et al. (1991). "Mechanism of activation of the vav protooncogene." *Cell Growth Differ* **2**(2): 95-105.
- Cowan, C. W., Y. R. Shao, et al. (2005). "Vav family GEFs link activated Ephs to endocytosis and axon guidance." *Neuron* **46**(2): 205-17.
- Crespo, P., K. E. Schuebel, et al. (1997). "Phosphotyrosine-dependent activation of Rac-1 GDP/GTP exchange by the vav proto-oncogene product." *Nature* **385**(6612): 169-72.
- Curto, E. V., H. N. Moseley, et al. (1996). "CORCEMA evaluation of the potential role of intermolecular transferred NOESY in the characterization of ligand-receptor complexes." *J Comput Aided Mol Des* **10**(5): 361-71.
- Das, B., X. Shu, et al. (2000). "Control of intramolecular interactions between the pleckstrin homology and Dbl homology domains of Vav and Sos1 regulates Rac binding." *J Biol Chem* **275**(20): 15074-81.
- De Sepulveda, P., K. Okkenhaug, et al. (1999). "Socs1 binds to multiple signalling proteins and suppresses steel factor-dependent proliferation." *EMBO J* **18**(4): 904-15.
- Decker, M., C. Moon, et al. (2005). "The immunological synapse and Rho GTPases." *Curr Top Microbiol Immunol* **291**: 61-90.
- Deckert, M., S. Tartare-Deckert, et al. (1996). "Functional and physical interactions of Syk family kinases with the Vav proto-oncogene product." *Immunity* **5**(6): 591-604.
- Deng, H., N. Zhadin, et al. (2001). "Dynamics of protein ligand binding on multiple time scales: NADH binding to lactate dehydrogenase." *Biochemistry* **40**(13): 3767-73.
- DiNitto, J. P., A. Delprato, et al. (2007). "Structural basis and mechanism of autoregulation in 3-phosphoinositide-dependent Grp1 family Arf GTPase exchange factors." *Mol Cell* **28**(4): 569-83.
- Dombroski, A. J., W. A. Walter, et al. (1992). "Polypeptides containing highly conserved regions of transcription initiation factor sigma 70 exhibit specificity of binding to promoter DNA." *Cell* **70**(3): 501-12.
- Dueber, J. E., B. J. Yeh, et al. (2003). "Reprogramming control of an allosteric signaling switch through modular recombination." *Science* **301**(5641): 1904-8.
- Eisenmesser, E. Z., D. A. Bosco, et al. (2002). "Enzyme dynamics during catalysis." *Science* **295**(5559): 1520-3.
- Eisenmesser, E. Z., O. Millet, et al. (2005). "Intrinsic dynamics of an enzyme underlies catalysis." *Nature* **438**(7064): 117-21.

- Eriksson, A. E., W. A. Baase, et al. (1992). "A cavity-containing mutant of T4 lysozyme is stabilized by buried benzene." *Nature* **355**(6358): 371-3.
- Faber, H. R. and B. W. Matthews (1990). "A mutant T4 lysozyme displays five different crystal conformations." *Nature* **348**(6298): 263-6.
- Fackler, O. T., W. Luo, et al. (1999). "Activation of Vav by Nef induces cytoskeletal rearrangements and downstream effector functions." *Molecular Cell* **3**(6): 729-739.
- Feher, V. A. and J. Cavanagh (1999). "Millisecond-timescale motions contribute to the function of the bacterial response regulator protein Spo0F." *Nature* **400**(6741): 289-93.
- Fernandez-Zapico, M. E., N. C. Gonzalez-Paz, et al. (2005). "Ectopic expression of VAV1 reveals an unexpected role in pancreatic cancer tumorigenesis." *Cancer Cell* **7**(1): 39-49.
- Fierke, C. A., K. A. Johnson, et al. (1987). "Construction and evaluation of the kinetic scheme associated with dihydrofolate reductase from *Escherichia coli*." *Biochemistry* **26**(13): 4085-92.
- Fischer, K. D., A. Zmuldzinas, et al. (1995). "Defective T-cell receptor signalling and positive selection of Vav-deficient CD4⁺ CD8⁺ thymocytes." *Nature* **374**(6521): 474-7.
- Frauenfelder, H., S. G. Sligar, et al. (1991). "The energy landscapes and motions of proteins." *Science* **254**(5038): 1598-603.
- Fujikawa, K., A. V. Miletic, et al. (2003). "Vav1/2/3-null mice define an essential role for Vav family proteins in lymphocyte development and activation but a differential requirement in MAPK signaling in T and B cells." *J Exp Med* **198**(10): 1595-608.
- Gaczynska, M. and P. A. Osmulski (2005). "Characterization of noncompetitive regulators of proteasome activity." *Methods Enzymol* **398**: 425-38.
- Gakidis, M. A., X. Cullere, et al. (2004). "Vav GEFs are required for beta2 integrin-dependent functions of neutrophils." *J Cell Biol* **166**(2): 273-82.
- Galland, F., S. Katzav, et al. (1992). "The products of the mcf-2 and vav proto-oncogenes and of the yeast gene cdc-24 share sequence similarities." *Oncogene* **7**(3): 585-7.
- Gladwin, M. T. and D. B. Kim-Shapiro (2008). "The functional nitrite reductase activity of the heme-globins." *Blood* **112**(7): 2636-47.
- Goodey, N. M. and S. J. Benkovic (2008). "Allosteric regulation and catalysis emerge via a common route." *Nat Chem Biol* **4**(8): 474-82.
- Gotoh, A., H. Takahira, et al. (1997). "Cross-linking of integrins induces tyrosine phosphorylation of the proto-oncogene product Vav and the protein tyrosine kinase Syk in human factor-dependent myeloid cells." *Cell Growth Differ* **8**(6): 721-9.

- Grey, M. J., C. Wang, et al. (2003). "Disulfide bond isomerization in basic pancreatic trypsin inhibitor: multisite chemical exchange quantified by CPMG relaxation dispersion and chemical shift modeling." J Am Chem Soc **125**(47): 14324-35.
- Groysman, M., C. S. Russek, et al. (2000). "Vav, a GDP/GTP nucleotide exchange factor, interacts with GDIs, proteins that inhibit GDP/GTP dissociation." FEBS Lett **467**(1): 75-80.
- Gulotta, M., H. Deng, et al. (2002). "Toward an understanding of the role of dynamics on enzymatic catalysis in lactate dehydrogenase." Biochemistry **41**(10): 3353-63.
- Gunasekaran, K., B. Ma, et al. (2004). "Is allostery an intrinsic property of all dynamic proteins?" Proteins **57**(3): 433-43.
- Gunther, U., T. Mittag, et al. (2002). "Probing Src homology 2 domain ligand interactions by differential line broadening." Biochemistry **41**(39): 11658-69.
- Hammes, G. G. (2002). "Multiple conformational changes in enzyme catalysis." Biochemistry **41**(26): 8221-8.
- Han, J., B. Das, et al. (1997). "Lck regulates Vav activation of members of the Rho family of GTPases." Mol Cell Biol **17**(3): 1346-53.
- Han, J., K. Luby-Phelps, et al. (1998). "Role of substrates and products of PI 3-kinase in regulating activation of Rac-related guanosine triphosphatases by Vav." Science **279**(5350): 558-60.
- Hantschel, O., B. Nagar, et al. (2003). "A myristoyl/phosphotyrosine switch regulates c-Abl." Cell **112**(6): 845-57.
- Hardy, J. A. and J. A. Wells (2004). "Searching for new allosteric sites in enzymes." Curr Opin Struct Biol **14**(6): 706-15.
- Heckert, N. A. (2003). "NIST Handbook 148: DATAPLOT Reference Manual." National Institute of Standards and Technology Handbook Series **1**.
- Henske, E. P., M. P. Short, et al. (1995). "Identification of VAV2 on 9q34 and its exclusion as the tuberous sclerosis gene TSC1." Ann Hum Genet **59**(Pt 1): 25-37.
- Hensmann, M., G. W. Booker, et al. (1994). "Phosphopeptide binding to the N-terminal SH2 domain of the p85 alpha subunit of PI 3'-kinase: a heteronuclear NMR study." Protein Sci **3**(7): 1020-30.
- Henzler-Wildman, K. and D. Kern (2007). "Dynamic personalities of proteins." Nature **450**(7172): 964-72.
- Henzler-Wildman, K. A., V. Thai, et al. (2007). "Intrinsic motions along an enzymatic reaction trajectory." Nature **450**(7171): 838-44.
- Heo, J., R. Thapar, et al. (2005). "Recognition and activation of Rho GTPases by Vav1 and Vav2 guanine nucleotide exchange factors." Biochemistry **44**(17): 6573-85.

- Hobert, O., B. Jallal, et al. (1996). "Interaction of Vav with ENX-1, a putative transcriptional regulator of homeobox gene expression." Mol Cell Biol **16**(6): 3066-73.
- Hofmann, T. G., S. P. Hehner, et al. (2000). "Caspase-dependent cleavage and inactivation of the Vav1 proto-oncogene product during apoptosis prevents IL-2 transcription." Oncogene **19**(9): 1153-63.
- Hornstein, I., A. Alcover, et al. (2004). "Vav proteins, masters of the world of cytoskeleton organization." Cell Signal **16**(1): 1-11.
- Huang, Z., S. Shiva, et al. (2005). "Enzymatic function of hemoglobin as a nitrite reductase that produces NO under allosteric control." J Clin Invest **115**(8): 2099-107.
- Jones, M. L., J. D. Craik, et al. (2004). "Regulation of SHP-1 tyrosine phosphatase in human platelets by serine phosphorylation at its C terminus." J Biol Chem **279**(39): 40475-83.
- Katzav, S. (2004). "Vav1: an oncogene that regulates specific transcriptional activation of T cells." Blood **103**(7): 2443-51.
- Katzav, S. (2007). "Flesh and blood: the story of Vav1, a gene that signals in hematopoietic cells but can be transforming in human malignancies." Cancer Lett **255**(2): 241-54.
- Katzav, S., J. L. Cleveland, et al. (1991). "Loss of the amino-terminal helix-loop-helix domain of the vav proto-oncogene activates its transforming potential." Mol Cell Biol **11**(4): 1912-20.
- Katzav, S., D. Martin-Zanca, et al. (1989). "vav, a novel human oncogene derived from a locus ubiquitously expressed in hematopoietic cells." EMBO J **8**(8): 2283-90.
- Kay, L. E. (1998). "Protein dynamics from NMR." Biochem Cell Biol **76**(2-3): 145-52.
- Kay, L. E. (1998). "Protein dynamics from NMR." Nat Struct Biol **5 Suppl**: 513-7.
- Kay, L. E. (2005). "NMR studies of protein structure and dynamics." J Magn Reson **173**(2): 193-207.
- Kern, D., E. Z. Eisenmesser, et al. (2005). "Enzyme dynamics during catalysis measured by NMR spectroscopy." Methods Enzymol **394**: 507-24.
- Kern, D. and E. R. Zuiderweg (2003). "The role of dynamics in allosteric regulation." Curr Opin Struct Biol **13**(6): 748-57.
- Kim, A. S., L. T. Kakalis, et al. (2000). "Autoinhibition and activation mechanisms of the Wiskott-Aldrich syndrome protein." Nature **404**(6774): 151-8.
- Kim, C., C. C. Marchal, et al. (2003). "The hemopoietic Rho/Rac guanine nucleotide exchange factor Vav1 regulates N-formyl-methionyl-leucyl-phenylalanine-activated neutrophil functions." J Immunol **171**(8): 4425-30.

- Kishimoto, T., T. Taga, et al. (1994). "Cytokine signal transduction." Cell **76**(2): 253-62.
- Knoetig, S. M., T. A. Torrey, et al. (2002). "CD19 signaling pathways play a major role for murine AIDS induction and progression." Journal of Immunology **169**(10): 5607-5614.
- Kolb, D. A. and G. Weber (1975). "Cooperativity of binding of anilinonaphthalenesulfonate to serum albumin induced by a second ligand." Biochemistry **14**(20): 4476-81.
- Kolb, D. A. and G. Weber (1975). "Quantitative demonstration of the reciprocity of ligand effects in the ternary complex of chicken heart lactate dehydrogenase with nicotinamide adenine dinucleotide oxalate." Biochemistry **14**(20): 4471-6.
- Korzhnev, D. M., I. Bezsonova, et al. (2006). "Probing the transition state ensemble of a protein folding reaction by pressure-dependent NMR relaxation dispersion." J Am Chem Soc **128**(15): 5262-9.
- Korzhnev, D. M., K. Kloiber, et al. (2004). "Probing slow dynamics in high molecular weight proteins by methyl-TROSY NMR spectroscopy: application to a 723-residue enzyme." J Am Chem Soc **126**(12): 3964-73.
- Korzhnev, D. M., P. Neudecker, et al. (2005). "Multiple-site exchange in proteins studied with a suite of six NMR relaxation dispersion experiments: an application to the folding of a Fyn SH3 domain mutant." J Am Chem Soc **127**(44): 15602-11.
- Korzhnev, D. M., V. Y. Orekhov, et al. (2003). "Off-resonance R1rho relaxation outside of the fast exchange limit: an experimental study of a cavity mutant of T4 lysozyme." J Biomol NMR **26**(1): 39-48.
- Korzhnev, D. M., X. Salvatella, et al. (2004). "Low-populated folding intermediates of Fyn SH3 characterized by relaxation dispersion NMR." Nature **430**(6999): 586-90.
- Kovrigina, E. L., J. G. Kempf, et al. (2006). "Faithful estimation of dynamics parameters from CPMG relaxation dispersion measurements." J Magn Reson **180**(1): 93-104.
- Kovrigina, E. L. and J. P. Loria (2006). "Enzyme dynamics along the reaction coordinate: critical role of a conserved residue." Biochemistry **45**(8): 2636-47.
- Lange, O. F., N. A. Lakomek, et al. (2008). "Recognition dynamics up to microseconds revealed from an RDC-derived ubiquitin ensemble in solution." Science **320**(5882): 1471-5.
- Leung, D. W. and M. K. Rosen (2005). "The nucleotide switch in Cdc42 modulates coupling between the GTPase-binding and allosteric equilibria of Wiskott-Aldrich syndrome protein." Proc Natl Acad Sci U S A **102**(16): 5685-90.

- Li, P., I. R. Martins, et al. (2008). "Internal dynamics control activation and activity of the autoinhibited Vav DH domain." Nat Struct Mol Biol **15**(6): 613-8.
- Lietha, D., X. Cai, et al. (2007). "Structural basis for the autoinhibition of focal adhesion kinase." Cell **129**(6): 1177-87.
- Lim, W. A. (2002). "The modular logic of signaling proteins: building allosteric switches from simple binding domains." Curr Opin Struct Biol **12**(1): 61-8.
- Llorca, O., E. Arias-Palomo, et al. (2005). "Global conformational rearrangements during the activation of the GDP/GTP exchange factor Vav3." EMBO J **24**(7): 1330-40.
- Lopez-Lago, M., H. Lee, et al. (2000). "Tyrosine phosphorylation mediates both activation and downmodulation of the biological activity of Vav." Mol Cell Biol **20**(5): 1678-91.
- Loria, J. P., R. B. Berlow, et al. (2008). "Characterization of enzyme motions by solution NMR relaxation dispersion." Acc Chem Res **41**(2): 214-21.
- Loria, J. P., M. Rance, et al. (1999). "A Relaxation-Compensated Carr-Purcell-Meiboom-Gill Sequence for Characterizing Chemical Exchange by NMR Spectroscopy." Journal of the American Chemical Society **121**(10): 2331-2332
- Loria, J. P., M. Rance, et al. (1999). "A TROSY CPMG sequence for characterizing chemical exchange in large proteins." J Biomol NMR **15**(2): 151-5.
- Lundstrom, P., P. Vallurupalli, et al. (2007). "A single-quantum methyl ¹³C-relaxation dispersion experiment with improved sensitivity." J Biomol NMR **38**(1): 79-88.
- Masterson, L. R., A. Mascioni, et al. (2008). "Allosteric cooperativity in protein kinase A." Proc Natl Acad Sci U S A **105**(2): 506-11.
- Miura-Shimura, Y., L. Duan, et al. (2003). "Cbl-mediated ubiquitinylation and negative regulation of Vav." J Biol Chem **278**(40): 38495-504.
- Moarefi, I., M. LaFevre-Bernt, et al. (1997). "Activation of the Src-family tyrosine kinase Hck by SH3 domain displacement." Nature **385**(6617): 650-3.
- Monod, J., J. P. Changeux, et al. (1963). "Allosteric proteins and cellular control systems." J Mol Biol **6**: 306-29.
- Monod, J., J. Wyman, et al. (1965). "ON THE NATURE OF ALLOSTERIC TRANSITIONS: A PLAUSIBLE MODEL." J Mol Biol **12**: 88-118.
- Mosteller, R., J. Han, et al. (2000). "Biochemical analysis of regulation of Vav, a guanine-nucleotide exchange factor for Rho family of GTPases." Methods Enzymol **325**: 38-51.

- Movilla, N. and X. R. Bustelo (1999). "Biological and regulatory properties of Vav-3, a new member of the Vav family of oncoproteins." Mol Cell Biol **19**(11): 7870-85.
- Mulder, F. A., B. Hon, et al. (2002). "Slow internal dynamics in proteins: application of NMR relaxation dispersion spectroscopy to methyl groups in a cavity mutant of T4 lysozyme." J Am Chem Soc **124**(7): 1443-51.
- Mulder, F. A., A. Mittermaier, et al. (2001). "Studying excited states of proteins by NMR spectroscopy." Nat Struct Biol **8**(11): 932-5.
- Mulder, F. A., N. R. Skrynnikov, et al. (2001). "Measurement of slow (micros-ms) time scale dynamics in protein side chains by (15)N relaxation dispersion NMR spectroscopy: application to Asn and Gln residues in a cavity mutant of T4 lysozyme." J Am Chem Soc **123**(5): 967-75.
- Muralidhara, B. K., S. S. Negi, et al. (2007). "Dissecting the thermodynamics and cooperativity of ligand binding in cytochrome P450eryF." J Am Chem Soc **129**(7): 2015-24.
- Nagar, B., O. Hantschel, et al. (2003). "Structural basis for the autoinhibition of c-Abl tyrosine kinase." Cell **112**(6): 859-71.
- Neudecker, P., D. M. Korzhnev, et al. (2006). "Assessment of the effects of increased relaxation dispersion data on the extraction of 3-site exchange parameters characterizing the unfolding of an SH3 domain." J Biomol NMR **34**(3): 129-35.
- Nicholson, K. L., M. Munson, et al. (1998). "Regulation of SNARE complex assembly by an N-terminal domain of the t-SNARE Sso1p." Nat Struct Biol **5**(9): 793-802.
- Nolz, J. C., T. S. Gomez, et al. (2005). "The Ezh2 methyltransferase complex: actin up in the cytosol." Trends Cell Biol **15**(10): 514-7.
- Ogilvy, S., A. G. Elefanty, et al. (1998). "Transcriptional regulation of vav, a gene expressed throughout the hematopoietic compartment." Blood **91**(2): 419-30.
- Paccani, S. R., M. Boncristiano, et al. (2005). "Defective Vav expression and impaired F-actin reorganization in a subset of patients with common variable immunodeficiency characterized by T-cell defects." Blood **106**(2): 626-634.
- Palmer, A. G., 3rd (2001). "Nmr probes of molecular dynamics: overview and comparison with other techniques." Annu Rev Biophys Biomol Struct **30**: 129-55.
- Palmer, A. G., 3rd (2004). "NMR characterization of the dynamics of biomacromolecules." Chem Rev **104**(8): 3623-40.
- Palmer, A. G., 3rd, M. J. Grey, et al. (2005). "Solution NMR spin relaxation methods for characterizing chemical exchange in high-molecular-weight systems." Methods Enzymol **394**: 430-65.

- Palmer, A. G., 3rd, C. D. Kroenke, et al. (2001). "Nuclear magnetic resonance methods for quantifying microsecond-to-millisecond motions in biological macromolecules." Methods Enzymol **339**: 204-38.
- Palmer, A. G., 3rd, C. D. Kroenke, et al. (2001). "Nuclear magnetic resonance methods for quantifying microsecond-to-millisecond motions in biological macromolecules." Methods Enzymol **339**: 204-38.
- Palmer, A. G., 3rd and F. Massi (2006). "Characterization of the dynamics of biomacromolecules using rotating-frame spin relaxation NMR spectroscopy." Chem Rev **106**(5): 1700-19.
- Pandey, A., A. V. Podtelejnikov, et al. (2000). "Analysis of receptor signaling pathways by mass spectrometry: identification of vav-2 as a substrate of the epidermal and platelet-derived growth factor receptors." Proc Natl Acad Sci U S A **97**(1): 179-84.
- Pawson, T. (2004). "Specificity in signal transduction: from phosphotyrosine-SH2 domain interactions to complex cellular systems." Cell **116**(2): 191-203.
- Perlman, M. E., D. G. Davis, et al. (1994). "Studies of inhibitor binding to Escherichia coli purine nucleoside phosphorylase using the transferred nuclear Overhauser effect and rotating-frame nuclear Overhauser enhancement." Biochemistry **33**(24): 7547-59.
- Pisegna, S., A. Zingoni, et al. (2002). "Src-dependent Syk activation controls CD69-mediated signaling and function on human NK cells." J Immunol **169**(1): 68-74.
- Pluskey, S., T. J. Wandless, et al. (1995). "Potent stimulation of SH-PTP2 phosphatase activity by simultaneous occupancy of both SH2 domains." J Biol Chem **270**(7): 2897-900.
- Popovych, N., S. Sun, et al. (2006). "Dynamically driven protein allostery." Nat Struct Mol Biol **13**(9): 831-8.
- Pufall, M. A. and B. J. Graves (2002). "Autoinhibitory domains: modular effectors of cellular regulation." Annu Rev Cell Dev Biol **18**: 421-62.
- Puil, L. and T. Pawson (1992). "Vagaries of vav." Curr Biol **2**(5): 275-7.
- Quaranta, M. G., B. Mattioli, et al. (2003). "HIV-1 Nef triggers Vav-mediated signaling pathway leading to functional and morphological differentiation of dendritic cells." Faseb J **17**(14): 2025-36.
- Rao, N., I. Dodge, et al. (2002). "The Cbl family of ubiquitin ligases: critical negative regulators of tyrosine kinase signaling in the immune system." J Leukoc Biol **71**(5): 753-63.
- Rick, S. W., J. W. Erickson, et al. (1998). "Reaction path and free energy calculations of the transition between alternate conformations of HIV-1 protease." Proteins **32**(1): 7-16.
- Rombel, I., A. North, et al. (1998). "The bacterial enhancer-binding protein NtrC as a molecular machine." Cold Spring Harb Symp Quant Biol **63**: 157-66.

- Schmitz, A. A., E. E. Govek, et al. (2000). "Rho GTPases: signaling, migration, and invasion." Exp Cell Res **261**(1): 1-12.
- Schuebel, K. E., X. R. Bustelo, et al. (1996). "Isolation and characterization of murine vav2, a member of the vav family of proto-oncogenes." Oncogene **13**(2): 363-71.
- Selzer, T., S. Albeck, et al. (2000). "Rational design of faster associating and tighter binding protein complexes." Nat Struct Biol **7**(7): 537-41.
- Simmons, A., B. Gangadharan, et al. (2005). "Nef-mediated lipid raft exclusion of UbcH7 inhibits Cbl activity in T cells to positively regulate signaling." Immunity **23**(6): 621-34.
- Skrynnikov, N. R., F. A. Mulder, et al. (2001). "Probing slow time scale dynamics at methyl-containing side chains in proteins by relaxation dispersion NMR measurements: application to methionine residues in a cavity mutant of T4 lysozyme." J Am Chem Soc **123**(19): 4556-66.
- Sorokina, E. M. and J. Chernoff (2005). "Rho-GTPases: new members, new pathways." J Cell Biochem **94**(2): 225-31.
- Sprangers, R., A. Gribun, et al. (2005). "Quantitative NMR spectroscopy of supramolecular complexes: dynamic side pores in ClpP are important for product release." Proc Natl Acad Sci U S A **102**(46): 16678-83.
- Stebbins, C. C., C. Watzl, et al. (2003). "Vav1 dephosphorylation by the tyrosine phosphatase SHP-1 as a mechanism for inhibition of cellular cytotoxicity." Mol Cell Biol **23**(17): 6291-9.
- Su, I. H., M. W. Dobenecker, et al. (2005). "Polycomb group protein ezh2 controls actin polymerization and cell signaling." Cell **121**(3): 425-36.
- Sugase, K., H. J. Dyson, et al. (2007). "Mechanism of coupled folding and binding of an intrinsically disordered protein." Nature **447**(7147): 1021-5.
- Tamas, P., Z. Solti, et al. (2003). "Mechanism of epidermal growth factor regulation of Vav2, a guanine nucleotide exchange factor for Rac." J Biol Chem **278**(7): 5163-71.
- Tamas, P., Z. Solti, et al. (2001). "Membrane-targeting is critical for the phosphorylation of Vav2 by activated EGF receptor." Cell Signal **13**(7): 475-81.
- Tarakhovsky, A., M. Turner, et al. (1995). "Defective antigen receptor-mediated proliferation of B and T cells in the absence of Vav." Nature **374**(6521): 467-70.
- Taylor, S. S., J. Yang, et al. (2004). "PKA: a portrait of protein kinase dynamics." Biochim Biophys Acta **1697**(1-2): 259-69.
- Tolkachev, D., P. Xu, et al. (2003). "Probing the kinetic landscape of transient peptide-protein interactions by use of peptide (15)n NMR relaxation dispersion spectroscopy: binding of an antithrombin peptide to human prothrombin." J Am Chem Soc **125**(41): 12432-42.

- Tollinger, M., N. R. Skrynnikov, et al. (2001). "Slow dynamics in folded and unfolded states of an SH3 domain." *J Am Chem Soc* **123**(46): 11341-52.
- Turner, M. and D. D. Billadeau (2002). "VAV proteins as signal integrators for multi-subunit immune-recognition receptors." *Nat Rev Immunol* **2**(7): 476-86.
- Tybulewicz, V. L. (2005). "Vav-family proteins in T-cell signalling." *Curr Opin Immunol* **17**(3): 267-74.
- Tybulewicz, V. L., L. Ardouin, et al. (2003). "Vav1: a key signal transducer downstream of the TCR." *Immunol Rev* **192**: 42-52.
- Tzeng, Y. L., V. A. Feher, et al. (1998). "Characterization of interactions between a two-component response regulator, Spo0F, and its phosphatase, RapB." *Biochemistry* **37**(47): 16538-45.
- Tzeng, Y. L. and J. A. Hoch (1997). "Molecular recognition in signal transduction: the interaction surfaces of the Spo0F response regulator with its cognate phosphorelay proteins revealed by alanine scanning mutagenesis." *J Mol Biol* **272**(2): 200-12.
- Umbarger, H. E. (1992). "The origin of a useful concept--feedback inhibition." *Protein Sci* **1**(10): 1392-5.
- Villaverde, A. (2003). "Allosteric enzymes as biosensors for molecular diagnosis." *FEBS Lett* **554**(1-2): 169-72.
- Volkman, B. F., D. Lipson, et al. (2001). "Two-state allosteric behavior in a single-domain signaling protein." *Science* **291**(5512): 2429-33.
- Wand, A. J. (2001). "Dynamic activation of protein function: a view emerging from NMR spectroscopy." *Nat Struct Biol* **8**(11): 926-31.
- Wang, L., Y. Pang, et al. (2001). "Functional dynamics in the active site of the ribonuclease binase." *Proc Natl Acad Sci U S A* **98**(14): 7684-9.
- Watt, E. D., H. Shimada, et al. (2007). "The mechanism of rate-limiting motions in enzyme function." *Proc Natl Acad Sci U S A* **104**(29): 11981-6.
- Weiss, M. and T. Nilsson (2003). "A kinetic proof-reading mechanism for protein sorting." *Traffic* **4**(2): 65-73.
- Woessner, D. E. (1961). "Nuclear Transfer Effects in Nuclear Magnetic Resonance Pulse Experiments " *The Journal of Chemical Physics* **35**(1): 41-7.
- Wolf-Watz, M., V. Thai, et al. (2004). "Linkage between dynamics and catalysis in a thermophilic-mesophilic enzyme pair." *Nat Struct Mol Biol* **11**(10): 945-9.
- Wolfenden, R. and M. J. Snider (2001). "The depth of chemical time and the power of enzymes as catalysts." *Acc Chem Res* **34**(12): 938-45.
- Wu, J., A. Katrekar, et al. (2006). "Identification of substrates of human protein-tyrosine phosphatase PTPN22." *J Biol Chem* **281**(16): 11002-10.

- Wu, J., D. G. Motto, et al. (1996). "Vav and SLP-76 interact and functionally cooperate in IL-2 gene activation." *Immunity* **4**(6): 593-602.
- Wulf, G. M., C. N. Adra, et al. (1993). "Inhibition of hematopoietic development from embryonic stem cells by antisense vav RNA." *EMBO J* **12**(13): 5065-74.
- Xu, W., A. Doshi, et al. (1999). "Crystal structures of c-Src reveal features of its autoinhibitory mechanism." *Mol Cell* **3**(5): 629-38.
- Xu, W., S. C. Harrison, et al. (1997). "Three-dimensional structure of the tyrosine kinase c-Src." *Nature* **385**(6617): 595-602.
- Yabana, N. and M. Shibuya (2002). "Adaptor protein APS binds the NH₂-terminal autoinhibitory domain of guanine nucleotide exchange factor Vav3 and augments its activity." *Oncogene* **21**(50): 7720-9.
- Yao, X., M. K. Rosen, et al. (2008). "Estimation of the available free energy in a LOV2-J alpha photoswitch." *Nat Chem Biol* **4**(8): 491-7.
- Yohe, M. E., K. Rossman, et al. (2008). "Role of the C-terminal SH3 domain and N-terminal tyrosine phosphorylation in regulation of Tim and related Dbl-family proteins." *Biochemistry* **47**(26): 6827-39.
- Yohe, M. E., K. L. Rossman, et al. (2007). "Auto-inhibition of the Dbl family protein Tim by an N-terminal helical motif." *J Biol Chem* **282**(18): 13813-23.
- Yon, J. M., D. Perahia, et al. (1998). "Conformational dynamics and enzyme activity." *Biochimie* **80**(1): 33-42.
- Zaffran, Y., O. Destaing, et al. (2001). "CD46/CD3 costimulation induces morphological changes of human T cells and activation of Vav, Rac, and extracellular signal-regulated kinase mitogen-activated protein kinase." *J Immunol* **167**(12): 6780-5.
- Zeng, L., P. Sachdev, et al. (2000). "Vav3 mediates receptor protein tyrosine kinase signaling, regulates GTPase activity, modulates cell morphology, and induces cell transformation." *Mol Cell Biol* **20**(24): 9212-24.
- Zhang, R., F. W. Alt, et al. (1995). "Defective signalling through the T- and B-cell antigen receptors in lymphoid cells lacking the vav proto-oncogene." *Nature* **374**(6521): 470-3.
- Zheng, Y. (2001). "Dbl family guanine nucleotide exchange factors." *Trends Biochem Sci* **26**(12): 724-32.
- Zhou, Z., J. Yin, et al. (2007). "The calponin homology domain of Vav1 associates with calmodulin and is prerequisite to T cell antigen receptor-induced calcium release in Jurkat T lymphocytes." *J Biol Chem* **282**(32): 23737-44.
- Zugaza, J. L., M. A. Lopez-Lago, et al. (2002). "Structural determinants for the biological activity of Vav proteins." *J Biol Chem* **277**(47): 45377-92.

Appendix 1 Mathematica code for generating relaxation dispersion data for four-state equilibrium

```

fourStateR2[kmatrix_, pmatrix_, dwmatrix_, R2matrix_, appliedField_,
staticField_, residue_, protein_, T_] := Module[{}],

MBmatrix = {
{-I*dwmatrix[[residue, 1]] 942.45 - kmatrix[[protein, 1]]
pmatrix[[protein, 2]]/(pmatrix[[protein, 1]] + pmatrix[[protein, 2]]) -
kmatrix[[protein, 4]] pmatrix[[protein, 4]]/(pmatrix[[protein, 1]] +
pmatrix[[protein, 4]]) - R2matrix[[residue, staticField, 1]], kmatrix[[protein, 1]]
pmatrix[[protein, 1]]/(pmatrix[[protein, 1]] + pmatrix[[protein, 2]]), 0,
kmatrix[[protein, 4]] pmatrix[[protein, 1]]/(pmatrix[[protein, 1]] +
pmatrix[[protein, 4]])},
{kmatrix[[protein, 1]] pmatrix[[protein, 2]]/(pmatrix[[protein, 1]] +
pmatrix[[protein, 2]]), -I*dwmatrix[[residue, 2]] 942.45 - kmatrix[[protein, 1]]
pmatrix[[protein, 1]]/(pmatrix[[protein, 1]] + pmatrix[[protein, 2]]) -
kmatrix[[protein, 2]] pmatrix[[protein, 3]]/(pmatrix[[protein, 2]] +
pmatrix[[protein, 3]]) -R2matrix[[residue, staticField, 2]], kmatrix[[protein, 2]]
pmatrix[[protein, 2]]/(pmatrix[[protein, 2]] + pmatrix[[protein, 3]]), 0},
{0, kmatrix[[protein, 2]] pmatrix[[protein, 3]]/(pmatrix[[protein, 2]] +
pmatrix[[protein, 3]]), -I*dwmatrix[[residue, 3]] 942.45 - kmatrix[[protein, 2]]
pmatrix[[protein, 2]]/(pmatrix[[protein, 2]] + pmatrix[[protein, 3]]) -
kmatrix[[protein, 3]] pmatrix[[protein, 4]]/(pmatrix[[protein, 3]]
+ pmatrix[[protein, 4]]) - R2matrix[[residue, staticField, 3]], kmatrix[[protein, 3]]
pmatrix[[protein, 3]]/(pmatrix[[protein, 3]] + pmatrix[[protein, 4]])},
{kmatrix[[protein, 4]] pmatrix[[protein, 4]]/(pmatrix[[protein, 1]] +
pmatrix[[protein, 4]]), 0, kmatrix[[protein, 3]] pmatrix[[protein,
4]]/(pmatrix[[protein, 3]] + pmatrix[[protein, 4]]), -I*dwmatrix[[residue, 4]]
942.45 - kmatrix[[protein, 3]] pmatrix[[protein, 3]]/(pmatrix[[protein, 3]] +
pmatrix[[protein, 4]]) - kmatrix[[protein, 4]] pmatrix[[protein,
1]]/(pmatrix[[protein, 1]] + pmatrix[[protein, 4]]) - R2matrix[[residue, staticField,
4]]}}};

MBmatrixstar = {
{I*dwmatrix[[residue, 1]] 942.45 - kmatrix[[protein, 1]]
pmatrix[[protein, 2]]/(pmatrix[[protein, 1]] + pmatrix[[protein, 2]]) -
kmatrix[[protein, 4]] pmatrix[[protein, 4]]/(pmatrix[[protein, 1]] +
pmatrix[[protein, 4]]) - R2matrix[[residue, staticField, 1]], kmatrix[[protein, 1]]
pmatrix[[protein, 1]]/(pmatrix[[protein, 1]] + pmatrix[[protein, 2]]), 0,
kmatrix[[protein, 4]] pmatrix[[protein, 1]]/(pmatrix[[protein, 1]] +
pmatrix[[protein, 4]])},

```

```

{kmatrix[[protein, 1]] pmatrix[[protein, 2]]/(pmatrix[[protein, 1]] +
pmatrix[[protein, 2]]), I*dwmatrix[[residue, 2]] 942.45 - kmatrix[[protein, 1]]
pmatrix[[protein, 1]]/(pmatrix[[protein, 1]] + pmatrix[[protein, 2]]) -
kmatrix[[protein, 2]] pmatrix[[protein, 3]]/(pmatrix[[protein, 2]] +
pmatrix[[protein, 3]]) -R2matrix[[residue, staticField, 2]], kmatrix[[protein, 2]]
pmatrix[[protein, 2]]/(pmatrix[[protein, 2]] + pmatrix[[protein, 3]]), 0},
{0, kmatrix[[protein, 2]] pmatrix[[protein, 3]]/(pmatrix[[protein, 2]] +
pmatrix[[protein, 3]]), I*dwmatrix[[residue, 3]] 942.45 - kmatrix[[protein, 2]]
pmatrix[[protein, 2]]/(pmatrix[[protein, 2]] + pmatrix[[protein, 3]]) -
kmatrix[[protein, 3]] pmatrix[[protein, 4]]/(pmatrix[[protein, 3]]
+pmatrix[[protein, 4]]) - R2matrix[[residue, staticField, 3]], kmatrix[[protein, 3]]
pmatrix[[protein, 3]]/(pmatrix[[protein, 3]] + pmatrix[[protein, 4]])},
{kmatrix[[protein, 4]] pmatrix[[protein, 4]]/(pmatrix[[protein, 1]] +
pmatrix[[protein, 4]]), 0, kmatrix[[protein, 3]] pmatrix[[protein,
4]]/(pmatrix[[protein, 3]] + pmatrix[[protein, 4]]), I*dwmatrix[[residue, 4]]
942.45 - kmatrix[[protein, 3]] pmatrix[[protein, 3]]/(pmatrix[[protein, 3]] +
pmatrix[[protein, 4]]) - kmatrix[[protein, 4]] pmatrix[[protein,
1]]/(pmatrix[[protein, 1]] + pmatrix[[protein, 4]]) - R2matrix[[residue, staticField,
4]]} };

```

```
base =
```

```
MatrixExp[MBmatrix*T/(8*appliedField)].MatrixExp[ MBmatrixstar*T/(8*appliedField)].MatrixExp[MBmatrixstar*T/(8*appliedField)].MatrixExp[MBmatrix*T/(8*appliedField)];
```

```
finalMBmatrix = MatrixPower[base, 2*appliedField];
```

```
populationMatrix = {
{pmatrix[[protein, 1]], 0, 0, 0},
{0, pmatrix[[protein, 2]], 0, 0},
{0, 0, pmatrix[[protein, 3]], 0},
{0, 0, 0, pmatrix[[protein, 4]]} };

```

```
finalMagnetization = finalMBmatrix.populationMatrix;
```

```
R2calculated = -(1/T)*Log[Re[Total[finalMagnetization, 2]]];
```

```
Return[R2calculated]
```

```
];
```

```
generalCalculation =
```

```
Function[{kmatrix, pmatrix, dwmatrix, R2matrix, appliedField, staticField,
residue, protein, T}, fourStateR2[kmatrix, pmatrix, dwmatrix, R2matrix,
appliedField, staticField, residue, protein, T]];

```

```

data = Table[
  Table[
    Table[
      Table[{i*2, 942.45, k, h, 0.04/(8*i),
temp = Apply[generalCalculation,
  {{{665.14, 665.14, 665.14, 1808.04}}},
  {{0.8, 0.07, 0.03, 0.1}},
  {{0.9, 0.0, 0.0, 0.6}},
  {{{20.0, 5.0, 5.0, 20.0}}},
  i, j, k, h, 0.04}], j},
    {i, 1, 20}],
  {j, 1, 1}],
  {k, 1, 1}],
  {h, 1, 1}];

```

(*In methamatica, spaces between variables stand for product operation.*)

Appendix 2 Step by step CPMG relaxation dispersion data analyses

Data directory:

Data files from the magnet computer normally including raw data file—*fid*, parameter file—*procp*, error, if any, and progress recording (experiment starting time and ending time, if PS command is on, time of each ps is finished) file—*log*, user's experimental notebook—*text*. Some NMR managers prefer to save a few more potentially useful files from the magnet computer, which include *conpar*, *global* and the c code for the pulse sequence. Nevertheless, *fid*, *procp*, *log* and *text* are the most basic files needed for further data analysis.

All data files from the magnet computer are stored in a subdirectory called **data.fid**. These files will be read only. The Perl scripts for data analysis are in the directory **ncyc**. Functions of the Perl scripts will be detailed below. The intermediate files generated during data analysis including *test.fid* and *test.ft* will be in directories with the name of **ncyc_ddd.fid** or **ncyc_ddd_2.fid** format (ddd are integer indexing the number of spin-echo element in the constant relaxation time period. ‘_2’ means the repeat of the proceeding ‘_ddd’). The files including *.nv* files, the calculated R_2 and all other files generated beyond these steps are also in **ncyc** or subdirectories within **ncyc**. These abovementioned directories, namely, **ncyc**, **data.fid**, **ncyc_ddd.fid** and **ncyc_ddd_2.fid**, are included in a directory specifying some details about the experiment, for example, **C13_CPMG_ADWT_fullncyc_15d_900M-120707.fid**.

Unwind interleaved data into individual HSQC

The CPMG data to be analyzed are all collected in an interleaved way by setting `array=phase,ncyc_cp`, `phase=1,2`, and `ncyc_cp=2, 40`. The order of fids collected and store is as following:

i of i*Δt	phase	ncyc_cp
0	1	2
0	1	40
0	2	2
0	2	40
1	1	2
1	1	40
1	2	2

1	2	40
.		
.		
.		
ni	1	2
ni	1	40
ni	2	2
ni	2	40

Therefore there are totally $ni \times \text{dimension of phase} \times \text{dimension of ncyc_cp}$ fids in a CPMG data set. In the example, for each ni , there are 4 (dimension of $\text{phase} \times \text{dimension of ncyc_cp}$) fids among which fid number 1 and 3 belong to ncyc_cp 2 and fid number 2 and 4 belong to ncyc_cp 40. Bearing this in mind, I wrote a perl script, called *get_spectra.pl* (Appendix 3), to unwind the interleaved fids into individual HSQC. A shell script called *get_spectra* is generated upon execution of *get_spectra.pl*. In order to do the job in the example, the *get_spectra* is as following:

```
#!/bin/csh
cut_fid_mf_redhat ../data.fid ../ncyc_002.fid 4 1 3 ncyc_cp
cut_fid_mf_redhat ../data.fid ../ncyc_040.fid 4 2 4 ncyc_cp
```

The first line `#!/bin/csh` is to invoke c shell in which the job is done. An executable file, *cut_fid_mf_redhat*, in **ncyc** executes the job. The location of *fid*, location and directory name of individual HSQC are the first and the second arguments for *cut_fid_mf_redhat*. The sixth and last argument is the inner most arraying parameter. In CPMG measurement, *ncyc* or *ncyc_cp* is normally the inner most array parameter. The third argument is how many fid for each t_1 increment. Among the fids for each t_1 increment, the two belong to the i^{th} *ncyc_cp* is the fourth and fifth arguments of *cut_fid_mf_redhat* on $i+1^{\text{th}}$ line. Upon execution of *get_spectra*, *ncyc_002.fid* and *ncyc_040.fid* will be generated, within each there are *fid*, *procpa*, and *text*, three files. In *ncyc_002.fid*, *fid* contains all and only fids for $\text{ncyc_cp} = 2$ and therefore is ready for normal process as a HSQC.

Besides cutting fids into individual HSQCs, there are also a few other functions in *get_spectra.pl*. For example, it writes the names of newly generated *ncyc_ddd.fid* and *ncyc_ddd_2.fid* into a file called *exp_name*. It will check whether there is error message in *log*. Finally it writes all *ncyc_cp* values in an ascending order into a file called *ncyccount*. Both *exp_name* and *ncyccount* will be used in subsequent data analysis.

Generate test.fid for each HSQC

Since each HSQC can be processed using identical *fid.com*, I wrote a Perl script, *fid_com.pl* (Appendix 4), to do so. It goes to the first **ncyc_ddd.fid** (normally **ncyc_000_2.fid**), in which command *varian* is executed and *fid.com* is saved interactively. Then *fid.com* is copied to each **ncyc_ddd(_2).fid** and executed to produce *test.fid* from *fid*.

Calibrate comb.com for generating test.ft from test.fid

A Perl script, *proc_com.pl* (Appendix 5), was written to generate appropriate nmrPipe scripts, *comb.com*, for further data analysis. In an interactive way, window boundaries, phasing and appropriate parameters for Gauss window function are calibrated. Both Gauss and Lorentz windows functions are included in *comb.com*. The Lorentz window function is a square bell function with offset 0.5 and end 0.95 in ¹H dimension and end 0.98 in ¹³C dimension.

Generate test.ft and .nv files for each HSQC

A Perl script, *process2DSpe.pl* (Appendix 6), was written to facilitate further data analysis using *comb.com*. Either Gauss or Lorentz window function can be selected to generate *test.ft* from *test.fid*. Subsequently *.nv* files will be generated from *test.ft*. While *test.fid* and *test.ft* are in **ncyc_ddd(_2).fid**, *.nv* files are in **ncyc/nvfiles**. The file *.nmrview* is generated to accommodate the newly generated *.nv* files so that the 'load state' command from 'file' menu can load all *.nv* files automatically.

Extracting peak intensity using Rate Analysis function of nmrview

These steps are for nmrview 5.0 and certain modifications may be needed for other versions of nmrview.

- 1) Type *./nv5* in **ncyc/nvfiles** and nmrview 5 will be invoked.
- 2) Click on File.
- 3) Click on Load State and all *.nv* files will be loaded.
- 4) Click on Windows.
- 5) Click on Add.
- 6) Input *ncyc_0* as window name and click create.
- 7) You can resize the window at your will.
- 8) Put the arrow within the window and right click. A menu will show up.
- 9) Click Attributes and the Attributes window will show up.
- 10) Click File and then click Dataset. A DatasetMngr window will pop up.
- 11) Double click on *ncyc_0.nv* and *ncyc_0.nv* will be added to the window *ncyc_0*.

- 12) Click Draw button on the Attribute window and the 2D spectrum will show up.
- 13) By clicking decrease and increase button of Lvl on Attributes window, you can adjust to make sure all the peaks of your interest visible.
- 14) Once again, put the arrow within the window and right click. The same manu will show up.
- 15) Click Peak and then click Pick and the Peak picking window will pop up.
- 16) Give it a name, ncyc_0 and click Pick. A peaklist with the name, ncyc_0, is generated. Every visible peak is picked.
- 17) Click Assign on main manu and then click Peaks. The window peakPanel is popped up. Click List and the click the peaklist, ncyc_0. The window peakPanel is renamed ncyc_0.
- 18) Click File on the peakPanel window and the click Write List.
- 19) The NMRView PeakList window is popped up and you can specify the peak list name to be saved. The peaklist has to be ended by .xpk in order for nmrview to recognize when it is later read into nmrview.
- 20) Click File on main manu and then click Save State.
- 21) Click Analysis on main manu and then click Rate Analysis to initiate Rate Analysis window.
- 22) Input ncyc_ as Matrix Root, nv as Matrix Extension. Use the default for Mode. Click Select to highlight the peaklist, ncyc_0. Click Setup to highlight time.txt.
- 23) Now dispersion curves can be inspected by pushing increase and decrease button of Peak. You can also go to peak number i by inputing i-1 and pressing increase button.
- 24) In the NMRView Console, first to activate PrintRateData by execute: source ~/nmrview/my_tcl/print_rate_data.tcl and then execute PrintRateData temp.
- 25) Then file *temp* is generated and saved in the folder **nvfiles**. Each peak on the peaklist has one and only one line in *temp*. Each line starts with peaklist name.peak number. For example, the first line of *temp* starts with ncyc_0.0 for peak number 0 on peaklist ncyc_0. The second element on line i is the index of the first HSQC and the intensity of the peak number i-1 on the first HSQC is the third element. Etc.
- 26) Eye inspection of *temp* to eliminate peaks of negative intensities because R_2 can not be calculated and also meaningless.

Calculating measured R_2 values of each peak

A Perl script, *R2calculation.pl* (Appendix 7), was written to calculate R_2 values based on *ncyc*count and the intensity file, *temp*. The output file is *R2app*. A Perl script, *inputgeneration.pl* (Appendix 8), was written to generate data files in the

right format for CPMG curve fitting program from *R2app*. Each CPMG data point is indexed by the model (model index), by the protein it belongs to (protein index), by the resonance it belongs to (resonance index), by the temperature at which it was collected (temperature index), by the static field strength at which it was collected at (field strength index), by the length of constant relaxation period (constant time index), and by the applied magnetic field strength at which it was collected (applied field strength index). The resulting data file is called *dynamic40_i* or *dynamic20_i* (i is indexing which protein the data belong to and at which temperature the data were collected. Table A1). All these data files are in a directory **datafiles** which is in the same directory where other executable files are.

Table A1
File indexes

Temperature	Field	ADwt	ADK208V	ADK208A	ADK208E
5	600	111 ^a	112	113	114
	800	121	122	123	124
	900	131	132	133	134
10	600	211	212	213	214
	800	221	222	223	224
	900	231	232	233	234
15	600	311	312	313	314
	800	321	322	323	324
	900	331	332	333	334

^a: Not all indexes have corresponding data collected. For example no data were collected on 900 MHz instrument at 5 °C. But this set of index surely covers all data collected.

The model indexes 31, 51, and 71 are for two-state, three-state, and four-state equilibrium, respectively. The protein index 1, 2, 3, and 4 are for ADWT, ADK208V, ADK208A, and ADK208E, respectively. ADK208E is always set to two-state equilibrium and AD proteins always share the same model, i.e. the same number of state, either two or three or four states. When AD proteins are assumed of three- or four-state equilibrium, on their thermodynamic boxes composed of all states, there is always one edge identical to the two-state equilibrium of ADK208E.

For all proteins at all temperatures, CPMG measurements were collected at the constant relaxation period, 40 ms (constant time index = 1).

Temperature index 1, 2, and 3 are for 5 °C, 10 °C, and 15 °C, respectively. When multiple temperatures are involved, there is an assumption that $\Delta\Omega$ s are temperature-invariant. In such a way, data across different temperatures can put constraints on one another. In a fast exchange situation as in AD proteins, this assumption is important for better fitting at higher temperatures. At higher temperatures, k_{ex} s are farther beyond $\Delta\Omega$ s and the exchange regime is in fast limit, in which $\Delta\Omega$ s and populations are not readily de-convolvable.

Field strength index 1, 2, and 3 are for 600 MHz, 800 MHz, and, 900MHz, respectively. When data from multiple fields are included, $\Delta\Omega$ s scale in proportion to field strengths.

Synthesized CPMG data are also of the same format as abovementioned.

Generate input data and parameter files for optimization

With the availability of all data files in directory **datafiles**, we can specify which data files are included in current curve fitting. These specified data files will be combined into a data file called *dynamic40* in the same directory where other executable files are. Afterwards we will have to specify which resonances are to be included in further data analysis. Data points belong to the specified resonances will be picked out of file *dynamic40* and written in the curve fitting input file, *cpmgdata*. This operation is accomplished using a perl script: *generatecpmgdata.pl* (Appendix 9).

Besides data input, constraints as well as initial guesses and boundaries for adjustable parameters are needed for further optimization. The information is recorded in the files, *kfile*, *pfile*, *wfile* and *rfile*. These files are generated using perl script: *curvefitting.pl* (Appendix 10).

There are 6*(maximum number of proteins, 5 in current circumstances) lines in *kfile* and *pfile*. The first 6 lines belong to protein 1 and the second 6 lines belong to protein 2, etc.. Among the six lines belong to one protein, the first, third and fifth lines are for dataset with relaxation time period, 40 ms, at 5 °C, 10 °C and 15 °C, respectively, while the second, fourth and sixth lines are for dataset with relaxation time period, 20 ms, at these three temperatures. There are 12 elements on each line in *kfiles* and *pfiles*. The first, second, third and fourth pair of the

elements on each line are initial guesses and initial uncertainties for k_{ex1} (p1), k_{ex2} (p2), k_{ex3} (p3), and k_{ex4} (p4), respectively. The last four elements are four fixing flags, f, and one flag for each k_{ex} or population.

There are $6 \times$ (maximum number of resonances, 31 in current circumstances) lines in *rfile*. The first 6 lines belong to resonance 1 and the second 6 lines belong to resonance 2, etc.. Among the six lines belong to one resonance, the first, third and fifth lines are for dataset with relaxation time period, 40 ms, at 5 °C, 10 °C and 15 °C, respectively, while the second, fourth and sixth lines are for dataset with relaxation time period, 20 ms, at these three temperatures. There are 18 elements on each line in *rfiles*. The first and second pairs of the elements on each line are initial guesses and initial uncertainties for R2C and R2O at 600 MHz, respectively. The next two pairs for R2C and R2O at 800 MHz, respectively. Etc. The last six elements are six fixing flags and one for each R2.

There are $2 \times$ (maximum number of resonances, 31 in current circumstances) lines in *wfile*. The first 2 lines belong to resonance 1 and the second 2 lines belong to resonance 2, etc.. Among the two lines belong to one resonance, the first line is for dataset with relaxation time period, 40 ms, and the second is for dataset with relaxation time period, 20 ms. There are 12 elements on each line in *wfiles*. The first, second, third and fourth pairs of the elements are initial guesses and initial uncertainties for Ω_{AC} , Ω_{AO} , Ω_{BO} , and Ω_{BC} , respectively. The last four elements are four fixing flags and one for each Ω .

These four files contains maximum number of possible given that the model of equilibrium is a four-state, all five proteins involved have measurements at all temperatures (5 °C, 10 °C, and 15 °C) with relaxation period at both 40 ms and 20 ms and all 31 resonances are included in the data analysis. The advantage of writing everything possible out is that location of a specific parameter in certain conditions is always fixed and therefore easier to track. These files will be modified according to what data are actually available in *cpmgdata*. For example, if data from protein 1 at 5 °C with relaxation period 40ms are contained in *cpmgdata*, the four fixing flags in the first line in *kfile* and *pfile* will be changed to unfixing flags, u. When all modifications to *kfile* and *pfile* are made according to availability of data, they will be modified again according to constraints. For example, $p_{AC} + p_{AO} + p_{BO} + p_{BC} = 1.0$ and therefore there are only three degrees of freedom among the four populations and the unfixing flag of p_{BC} is changed back to a fixing flag. The AO-BO edges of Proteins 1-4 are identical to the two-state equilibrium of DH. Therefore k_{ex} s of the edges of proteins 1-4 can be constrained upon k_{ex} of DH and their unfixing flags will be changed back to fixing ones. Similarly population of DH protein has one degree of freedom and

each of proteins 1-4 has a population can be constrained on population of DH and their unfixing flags will be changed back to fixing ones.

A parameter with an unfixing flag (u) is an adjustable one during optimization. A parameter with a fixing flag (f) can be a parameter with constraint from other adjustable parameters or a parameter is not in use. Although the whole content of *kfile*, *pfile*, *rfile*, and *wfile* will be read into the optimization program, only adjustable parameters will be varied systematically for minimization and parameters not in use will be ignored. In addition, only adjustable parameters will be assigned with lower and higher boundaries.

Optimization strategy

Once *cpmgdata*, *kfile*, *pfile*, *wfile* and *rfile* are ready to go, we can start the optimization process. First of all, data (experimental measurements and uncertainties) are read into the program (command line 1) and followed by parameters (including initial guesses) and other constraints (command line 2-5). For adjustable parameters, the low and high boundaries of them will be assigned when they are read into the program. The commands are more or less like the following after starting *cpmg_fit_linux8*:

```
> read cpmgdata d (1)
> read kfile k (2)
> read pfile p (3)
> read wfile w (4)
> read rfile R (5)
> min results (6)
```

These lines of command can also be written in a plain text file, for example, *run.txt*, without the promote symbol, >. The commands can be executed by typing *cpmg_fit_linux8 < run.txt*. It is handy using script to control the optimization process if there are a lot of commands or the same set of commands has to be executed repetitively.

There are several ways to do the curve fitting. The first way is to use simulated annealing. The method is robust if there is a defined global minimum. But it is time consuming and can be prohibited for problems with hundreds of adjustable parameters and thousands or tens of thousands of data points. The running time is linearly proportional to the number of data point and quadratically dependent on the number of adjustable parameters. In our current situation, a method solely based on simulated annealing is prohibitive.

Another way is to use gradient-decent-based algorithm as Lewis Kay and coworkers did, who strategically avoided mentioning which algorithms are used in curve fitting. The most potential drawback of this algorithm is that it really has no distinction between local minimum and global minimum. The latter can be elusive in real situation where the models are very complicated. Multiple curve fitting trials from different initial points can alleviate this issue if a single solution is obtained. However, chances are that multiple solutions will be obtained in these exercises. Once again, if there are finite numbers of well populated solutions, the one(s) with the lowest ChiSquare values are most likely the solution. The more solutions exist, the landscape of ChiSquare surface is more rough, the less certain we are about the solution. To demonstrate presence of finite number of solution is to try more initial points and no new solution found. Alternative way to demonstrate the presence of finite number of solutions is to define regions belonging to the same valley and such regions are covering the whole parameter space. It could be time-consuming but worth trying though. The best way of doing so is to synthesize some data and go from there because the input parameters are known.

Upon optimization initiation (command line 6), adjustable parameters are systematically changed as defined in the optimization modules using the gradient-descent-based one, `dnls.f`. One R2 corresponding to each and every data is calculated at current parameters (population, R2 nulls, `kexs`, `deltaOmega`) using modified Bloch equation. Then target ChiSquare function derived from all data is evaluated. Decisions about where to move and how to move are made by comparing current target function value with previous value(s).

Once termination criteria are met, the optimization is done and the ultimate outcome and other intermediate results will be written in the file *results*. Successful optimization means improvement of current target function value or changes of current parameter vector are smaller than pre-defined thresholds. An optimization will also be terminated if number of target function evaluation exceeds maximum number function evaluation specified. In such circumstances, the optimization is unsuccessful.

The C-code for global optimization using gradient-decent based method is in Appendix H, which was modified from c-code provided by Drs. Dmitry Korznev and Lewis Kay. In Appendix H, upon success of gradient-decent optimization, covariance matrix of all parameters is calculated using `dcov.f` module and square root of diagonal elements are standard deviation of each parameter.

Curve fitting of ADK208E data

Data from ADK208E domain protein at all three temperatures were fitted globally to a two-state equilibrium assuming that chemical shift differences are temperature invariant. The resulting k_{ex} s and p_A/p_B ratios will be used as constraints for k_{ex2} and p_{AO}/p_{BO} of AD proteins (wild type and K208A), respectively.

There will be one k_{ex} and population per protein per temperature. The second population will be calculated within R2_2_sites.c. There will be one R_2^0 per resonance per field per temperature. There will be one $\Delta\omega$ per resonance, which is assumed temperature-invariant. One set of resonances including all resonances showing relaxation dispersion in ADK208E. A second set of resonances including only the ones to be used later in four-state model. It is done and robust enough as predicted since it is two-state equilibrium anyway. The major species is populated above 90% and will be designate A state in ADK208E. Therefore in AD proteins, AO will be the major species on AO-BO transition. The remaining question is what is p_{AC}/p_{BC} . The difference between these two is the energetic coupling strength in whatever unit.

Some assumptions

Assuming the AC-BC edge is not perturbed by mutation and therefore $p_{AC}/p_{BC}(\text{wild type AD}) = p_{AC}/p_{BC}(\text{ADK208A})$. Such assumption is consistent with the observation that both front dynamic process resonances and ADK208E dynamic process resonances generate similar open population in the NSMB paper. By doing so, we have one population adjustable parameter per protein per temperature. Degeneracy due to populations will be completely eliminated since the closed edges of AD mutants (K208A) are the minor edge in comparison with the open edges. In addition, k_{ex4} of AD wild type is also shared with the AD mutants.

Appendix 3 get_spectra.pl

```
#!/usr/bin/perl -w

#
# This script will generate get_spectra and cut spectra into 2D automatically
# The original data directory is data.fid.
#

$currentpath = `pwd`;
chomp($currentpath);
$currentpath =~ /(.)\./+$/s;
$parentpath = $1;
$currentpath =~ /.+\/(.)$/s;

$datadirectory = "data.fid";
$directory = $parentpath."/".$datadirectory;
chdir $directory;

open (PAR, "proctar") or die "\nCan not open proctar.\n";
$a = 0;
while ($line = <PAR>) {
    @int = split (" ", $line);
    if ($int[0] eq "array") {
        $a++;
    }
    if ($a == 1 and $int[0] == 1) {
        if ($int[1] eq "phase,ncyc") {
            $array = "ncyc";
        }
        elsif ($int[1] eq "phase,ncyc_cp") {
            $array = "ncyc_cp";
        }
        else {print "\nI do not know which experiment is this.\n";}
        $a++;
    }
}
close(PAR);

open (PAR, "proctar") or die "\nCan not open proctar.\n";
open (OUTPUT, ">temp_get_spectra");
```

```

print OUTPUT "#!/bin/csh\n";

$a = 0;
while ($line = <PAR>) {
    @int = split (" ", $line);
    if ($int[0] eq $array) {
        $a++;
    }
    if ($a == 1 and $int[0]=~ ^d+/) {
        foreach $i (1..$int[0]){
            $j = $i + $int[0];
            $k = 2*$int[0];
            if ($i > 0) {
                $c = 0;
                foreach $b (1..$i-1) {
                    if ($int[$i] == $int[$b]) {$c++;}
                }
                if ($c == 0) {
                    if ($int[$i] < 10) {print OUTPUT
"cut_fid_mf_redhat ../data.fid ../ncyc_00$int[$i].fid $k $i $j $array\n";}
                    elsif ($int[$i] >= 10 and $int[$i] < 100) {print OUTPUT
"cut_fid_mf_redhat ../data.fid ../ncyc_0$int[$i].fid $k $i $j $array\n";}
                    else {print OUTPUT
"cut_fid_mf_redhat ../data.fid ../ncyc_$int[$i].fid $k $i $j $array\n";}
                }
                elsif ($c == 1) {
                    if ($int[$i] < 10) {print OUTPUT
"cut_fid_mf_redhat ../data.fid ../ncyc_00$int[$i]_2.fid $k $i $j $array\n";}
                    elsif ($int[$i] >= 10 and $int[$i] < 100) {print OUTPUT
"cut_fid_mf_redhat ../data.fid ../ncyc_0$int[$i]_2.fid $k $i $j $array\n";}
                    else {print OUTPUT
"cut_fid_mf_redhat ../data.fid ../ncyc_$int[$i]_2.fid $k $i $j $array\n";}
                }
                else {die "\nAre you sure that there are more than
two replications for one point?\n";}
            }
            else {
                if ($int[$i] < 10) {print OUTPUT
"cut_fid_mf_redhat ../data.fid ../ncyc_00$int[$i].fid $k $i $j $array\n";}
                elsif ($int[$i] >= 10 and $int[$i] < 100) {print
OUTPUT "cut_fid_mf_redhat ../data.fid ../ncyc_0$int[$i].fid $k $i $j $array\n";}
            }
        }
    }
}

```

```

                                else {print OUTPUT
"cut_fid_mf_redhat ../data.fid ../ncyc_$int[$i].fid $k $i $j $array\n";}
                                }
                                }
                                $a++;
                                }
                                }
close (PAR);
close (OUTPUT);
`chmod +x temp_get_spectra`;
`mv temp_get_spectra ../ncyc/get_spectra`;
chdir $currentpath;
`get_spectra`;

# The following operation is to write directory names of individual 2D HSQC into
a file called exp_names
# Determine the name of the directories
$currentpath = `pwd`;
chomp($currentpath);
$currentpath =~ /(.)\./s;
$parentpath = $1;
$currentpath =~ /.+\.(.+)/s;

# Generate the file exp_names
chdir $parentpath;
`ls -d *.fid > exp_names`;
`mv exp_names $currentpath`;
chdir $currentpath;

# Delete the line for data.fid
open (NAMES, "exp_names") or die "\nCan not open exp_names.\n";
open (OUTPUT, ">temp_exp_names");
while ($line = <NAMES>) {
    chomp ($line);
    if ($line ne "data.fid") {print OUTPUT "$line\n";}
}
close (NAMES);
close (OUTPUT);
`mv temp_exp_names exp_names`;

```

```

# Determine whether the folder contains every necessary file from spectrometer
computer
open (NAMES, "exp_names") or die "\nCan not open exp_names.\n";
chdir $parentpath;

$a = 0;
while ($line = <NAMES>) {
    chomp ($line);
    $directory = $parentpath."/".$line;
    chdir $directory;
    if (!-e "fid" or !-e "procpur" or !-e "text") {
        print "\n$line is not complete.\n";
        $a++;
    }
}
if ($a > 0) {print "\nSome of the experiment is not complete.\n";}
close (NAMES);

# Determine whether there is ADC overflow
chdir $currentpath;
$currentpath = `pwd`;
chomp($currentpath);
$currentpath =~ /(.)\./s;
$parentpath = $1;
$currentpath =~ /.\./s;

$datadirectory = "data.fid";
$directory = $parentpath."/".$datadirectory;
chdir $directory;

open (LOG, "log") or die "\nCan not opne log.\n";
$b = 0; $c = 0; $d = 0;
while ($line1 = <LOG>) {
    @int = split (" ", $line1);
    if ($int[6] eq "started") {$b++;}
    if ($int[6] eq "complete") {$c++;}
    if ($int[0] eq "ADC") {$d++;}
}

if ($b == 1 and $c == 1 and $d == 0) {print "\nAll experiments seems to be
OK.\n";}

```

```

elseif ($b == 1 and $c == 0 and $d == 0) {die "\n$line is not complete.\n";}
elseif ($b == 1 and $c == 1 and $d > 0) {die "\n$line ADC overflow.\n";}
else {print "\nsomething wrong.\n";}

close(LOG);

chdir $currentpath;
open (OUTPUT, ">ncyccount");

$a = 0;
chdir $directory;
open (PAR, "proctpar") or die "\nCan not open proctpar.\n";
while ($line = <PAR>) {
    @int = split(" ", $line);
    if ($int[0] eq $array) {$a++;}
    if ($a == 1 and $int[0] =~ ^d+/) {
        $a++;
        foreach $i (1..$int[0]) {
            foreach $j ($i+1..$int[0]) {
                if ($int[$int[0]-$i+1] <= $int[$int[0]-$j+1]) {
                    $stemp = $int[$int[0]-$i+1];
                    $int[$int[0]-$i+1] = $int[$int[0]-$j+1];
                    $int[$int[0]-$j+1] = $stemp;
                }
            }
        }
        foreach $i (1..$int[0]) {print OUTPUT "$int[$i] ";}
    }
}
close (PAR);

close (OUTPUT);
chdir $currentpath;

# delete get_spectra since executing get_spectra.pl is not time consuming anyway
`rm ./get_spectra`;

print "\nEOF, execute fid_com.pl\n";

```

Appendix 4 fid_com.pl

```

#!/usr/bin/perl -w

#
# This script is to generate fid.com and execute it in every folder
#

if (!-e "fid.com") {
    # Check whether exp_names.pl is executed or not since its output is
    # necessary for this script
    print "\nAre you sure that exp_names.pl has been executed?\n";
    print "\nInput y and n for yes and no, respectively:\n";
    $arg = <STDIN>;
    chomp ($arg);
    if ($arg eq "n") {die "\nExecute exp_names.pl first please.\n";}
    else {print "\nWell, let's continue.\n";}

    # Check whether the output file from exp_names.pl is openable
    open (NAMES, "exp_names") or die "\nCan not open exp_names.\n";
    @line = <NAMES>;
    chomp ($line[0]);

    # Determine the directory to generate fid.com
    $currentpath = `pwd`;
    chomp($currentpath);
    $currentpath =~ /(.)\./+$/s;
    $parentpath = $1;
    $currentpath =~ /.+\/(.)$/s;

    $directory = $parentpath."/".$line[0];

    close (NAMES);
    chdir $directory;

    # Generate fid.com
    `varian`;

    # Delete the line of sleep 5
    open(FIDCOM, "fid.com") or die "\nCan not open fid.com\n";
    open(OUTPUT, ">temp");

```

```

while ($line=<FIDCOM>){
    chomp($line);
    if($line ne 'sleep 5'){
        print(OUTPUT $line);
        print(OUTPUT "\n");
    }
}
close(OUTPUT);
close(FIDCOM);
`mv temp fid.com`;
`chmod +x fid.com`;
# Copy fid.com into parent directory
`cp fid.com $currentpath`;
}
else {
    $currentpath = `pwd`;
    chomp($currentpath);
    $currentpath =~ /(.)\./s;
    $parentpath = $1;
    $currentpath =~ /.+\/(.)\./s;
}

chdir $currentpath;
open (NAMES, "exp_names") or die "\nCan not open exp_names.\n";

# Copy fid.com into each data directory and execute therein
while ($line = <NAMES>) {
    chomp ($line);
    $directory = $parentpath."/".$line;
    `cp fid.com $directory`;
    chdir $directory;
    `chmod +x fid.com`;
    `fid.com`;
    `rm fid.com`;
    chdir $currentpath;
}

print "\nEOF, execute proc_com.pl\n";

```

Appendix 5 proc_com.pl

```
#!/usr/bin/perl -w

#
# This script is going to write the preliminary proc.com for further refinement
# It also guides the further refinement of this proc.com
#

if (-e "comb.com") {
    print "\ncomb.com exists! Do you want to proceed?\n";
    print "\nInput y and n for yes and no, respectively:\n";
    $arg = <STDIN>;
    chomp ($arg);
    if ($arg eq "n") {die "\nExecute process2Dspe.pl instead.\n";}
    else {print "\nWell then, let's continue.\n";}
}

#
# First of all, the program generate the file proc.com.
#

# Check whether fid_com.pl is executed or not since its output, test.fid is
# necessary for this script
print "\nAre you sure that fid_com.pl has been executed?\n";
print "\nInput y and n for yes and no, respectively:\n";
$arg = <STDIN>;
chomp ($arg);
if ($arg eq "n") {die "\nExecute fid_com.pl first please.\n";}
else {print "\nWell, let's continue.\n";}

# Check whether the output file from fid_com.pl is present
open (NAMES, "exp_names") or die "\nCan not open exp_names.\n";
$a = 0;
$b = 0;

$currentpath = `pwd`;
chomp($currentpath);
$currentpath =~ /(.)\./+$/s;
$parentpath = $1;
```

```

$currentpath =~ /\.+\(.\+)/s;

while ($line = <NAMES>) {
    chomp ($line);
    $directory = $parentpath."/".$line;

    # Remember the first directory for further usage
    if ($b == 0) {
        $directory1 = $directory;
        $b++;
    }

    chdir $directory;

    if (!-e "test.fid") {$a++;}
}
if ($a > 0) {die "\nfid.com is not executed in every data folder.\n";}
close (NAMES);

# Go to the first data folder to write and refine proc.com.
chdir $directory1;

#
# Call subroutine fl180_finding to find out whether the indirect mention is half
# dwell time delay or not
# The script assume all the spectra have the same fl180 value.
#
fl180_finding();
#####
sub fl180_finding {
    open (PARA, "proctpar") or die "\nCan not open proctpar.\n";
    $j = 0; $k = 0;
    if ($k == 0) {
        while ($line = <PARA>) {
            @int = split(" ", $line);
            if ($int[0] eq "fl180") {foreach $i (1..10) {if ($int[$i] =
~\d+/) {$j++;}}}
            if ($j == 10 and $int[0] == 1 and $k == 0) {
                $k++;
                $fl180 = $int[1];
            }
        }
    }
}

```

```

    }
  }
  close (PARA);
  return;
}
#####

# Write proc.com with the right setting according to f1180 value

print "\nWrite proc.com with the right setting according to f1180 value\n";
print "\nThe GM is commented out for both dimentions.\n";

if (-e "proc.com") {`rm ./proc.com`;}
open (OUTPUT, ">proc.com");

print OUTPUT "#!/bin/csh\n";
print OUTPUT "nmrPipe -in test.fid          \\n";
print OUTPUT "| nmrPipe -fn POLY -time          \\n";
print OUTPUT "| nmrPipe -fn SP -off 0.5 -end 0.95 -pow 2 -c 1.0    \\n";
print OUTPUT "#| nmrPipe -fn GM -g1 17 -g2 22 -g3 0.0          \\n";
print OUTPUT "| nmrPipe -fn ZF -zf 1 -auto          \\n";
print OUTPUT "| nmrPipe -fn FT -verb          \\n";
print OUTPUT "| nmrPipe -fn PS -p0 198.0 -p1 0.0 -di          \\n";
print OUTPUT "| nmrPipe -fn EXT -sw          \\n";
print OUTPUT "| nmrPipe -fn TP          \\n";

if ($f1180 eq "n") {print OUTPUT "| nmrPipe -fn SP -off 0.5 -end 0.98 -pow 2 -
c 0.5    \\n";}
elseif ($f1180 eq "y") {print OUTPUT "| nmrPipe -fn SP -off 0.5 -end 0.98 -pow
2 -c 1.0    \\n";}

print OUTPUT "#| nmrPipe -fn GM -g1 20 -g2 30 -g3 0.0          \\n";
print OUTPUT "| nmrPipe -fn ZF -zf 1 -auto          \\n";
print OUTPUT "| nmrPipe -fn FT -verb          \\n";

if ($f1180 eq "n") {print OUTPUT "| nmrPipe -fn PS -p0 0.0 -p1 0.0 -di
\\n";}
elseif ($f1180 eq "y") {print OUTPUT "| nmrPipe -fn PS -p0 -90 -p1 180.0 -di
\\n";}

```

```

print OUTPUT "| nmrPipe -fn POLY -auto -ord 1          \\n";
print OUTPUT "| nmrPipe -fn TP                        \\n";
print OUTPUT "| nmrPipe -fn POLY -auto -ord 1          \\n";
print OUTPUT " -out test.ft -verb 2 -ov\\n";
close (OUTPUT);

`chmod +x proc.com`;

# Specify the edges of the spectra if desirable
print "\\nproc.com is going to be executed and the test.ft will generated for
preliminary inspection.\\n";
print "\\nPlease just focus on the window of the spectra to determine whether you
want to narrow the window or not.\\n";
print "\\nIf you do want to narrow the window, remember the edges in ppm.\\n";
print "\\nPress ENTER to continue:\\n";
$arg = <STDIN>;
chomp ($arg);

if ($arg eq "") {
    `proc.com`;
    `nmrDraw`;
}

print "\\nDo you want to change the spectra window?\\n";
print "\\nInput y and n for yes and no, respectively:\\n";
$arg = <STDIN>;
chomp ($arg);
if ($arg eq "y") {
    print "\\nPlease input the edge of the upfield (right side) in ppm with ppm
attached to the number without space.\\n";
    print "\\nExamples, -1.0ppm, 3.9ppm etc.\\n";
    $x1 = <STDIN>;
    chomp ($x1);

    print "\\nPlease input the edge of the downfield (left side) in ppm with ppm
attached to the number without space.\\n";
    $xn = <STDIN>;
    chomp ($xn);

    rewrite_proc_com1();
}

```

```

# Phasing the spectra
print "\nNext step is to phase the spectra.\n";
print "\nSince it is always a good idea to inspect the spectra, this step is performed
without asking.\n";

$arg = "n";
while ($arg eq "n") {

    `proc.com`;
    `nmrDraw`;

    print "\nDo you think the phasing is OK?\n";
    print "\nInput y and press RETURN for answering yes or Input n and press
RETURN for answering no, please:\n";          $arg = <STDIN>;
    chomp ($arg);

    if ($arg eq "n") {
        print "\nThe changes have to be made on the phasing. Input change
in p0 in degree:\n";
        $arg1 = <STDIN>;
        chomp($arg1);
        $p0 = $arg1;

        print "\nInput change in p1 in degree:\n";
        $arg2 = <STDIN>;
        chomp($arg2);
        $p1 = $arg2;
        rewrite_proc_com2();
    }
}

# Calibrate g1 and g2 for both dimentions
# Rewrite proc.com to comment out SP and GM for both dimentions

rewrite_proc_com3();

# Process 2D spectra without any window funtions
`proc.com`;
# Pick AND edit peak list

```

```

print "\nBoth SP and GM are commented out in proc.com and the test.ft is
generated.\n";
print "\nPlease peak a reasonable peak list in the next step.\n";
print "\nThe name of the peaklist is test.tab.\n";
print "\nPress ENTER to proceed:\n";

$arg = <STDIN>;
chomp ($arg);

if ($arg eq "") {\`nmrDraw`;}

# Determine g1 values for both dimentions
g1_finding();
rewrite_proc_com4();

# Calibrate g2
print "\nCheck whether g2 is OK.\n";
print "\nPress ENTER to proceed\n";
$arg = <STDIN>;

$arg = "n";
while ($arg eq "n") {

    \`proc.com`;
    \`nmrDraw`;

    print "\nDo you think the g2 values for both dimentions are OK?\n";
    print "\nInput y and press RETURN for answering yes or Input n and press
RETURN for answering no, please:\n";
    $arg = <STDIN>;
    chomp ($arg);
    if ($arg eq "n") {
        print "\nInput g2 value for proton dimention:\n";
        $arg1 = <STDIN>;
        chomp($arg1);
        $g2_H = $arg1;

        print "\nInput g2 value for X dimention:\n";
        $arg2 = <STDIN>;
        chomp($arg2);
        $g2_X = $arg2;
    }
}

```

```

        rewrite_proc_com5();
    }
    elsif ($arg eq "y") {};
    else {die "\nNo idea what you want.\n";}
}

```

Move proc.com into the master folder for further usage. At this point, proc.com is supposed to be well refined.

```

`mv proc.com $currentpath`;
chdir $currentpath;

```

Check the availability of proc.com in the master folder.

```

if (-e "proc.com") {
    rewrite_proc_com6();
    print "\nNow proc.com is calibrated in $directory1 and is moved into
$currentpath.\n";
    print "\nproc.com is moved into comb.com.\n";
    `mv proc.com comb.com`;
}

```

```
#####
```

```

sub g1_finding {

    if (-e "test.tab") {$table = "test.tab";}
    else {die "\ntest.tab does not exist.\n";}

    open (TABLE, $table);
    $save_H = 0;
    $save_X = 0;
    $i = 0;
    while ($line = <TABLE>) {
        @int = split (" ", $line);
        if ($int[0] =~ ^\d+/) {
            $save_H = $save_H + $int[11];
            $save_X = $save_X + $int[12];
            $i++;
        }
    }
    $g1_H = $save_H/$i;
    $g1_X = $save_X/$i;
    close (TABLE);
}

```

```

        return;
    }
#####
sub rewrite_proc_com5 {
    open (PROC, "proc.com") or die "\nCan not open proc.com.\n";
    if (-e "temp") {`rm ./temp`;}
    open (OUTPUT, ">temp");

    $k = 0;
    while ($line = <PROC>) {
        @int = split (" ", $line);
        if ($k == 0 and $int[3] eq "GM") {
            print OUTPUT "| nmrPipe -fn GM -g1 $g1_H -g2 $g2_H -g3 0.0
\\n";
            $k++;
        }
        elsif ($k == 1 and $int[3] eq "GM") {
            print OUTPUT "| nmrPipe -fn GM -g1 $g1_X -g2 $g2_X -g3 0.0
\\n";
        }
        else {print OUTPUT $line;}
    }
    `mv temp proc.com`; `chmod +x proc.com`;
    close (PROC);
    close (OUTPUT);
    return;
}
#####
sub rewrite_proc_com4 {
    open (PROC, "proc.com") or die "\nCan not open proc.com.\n";
    if (-e "temp") {`rm ./temp`;}
    open (OUTPUT, ">temp");

    $k = 0;
    while ($line = <PROC>) {
        @int = split (" ", $line);
        if ($k == 0 and $int[3] eq "GM") {
            $g2_H = 1.5*$g1_H;
            print OUTPUT "| nmrPipe -fn GM -g1 $g1_H -g2 $g2_H -
g3 0.0    \\n";
            $k++;
        }
    }
}

```

```

    }
    elseif ($k == 1 and $int[3] eq "GM") {
        $g2_X = 1.5*$g1_X;
        print OUTPUT "| nmrPipe -fn GM -g1 $g1_X -g2 $g2_X -
g3 0.0    \\n";
    }
    else {print OUTPUT $line;}
}
`mv temp proc.com`; `chmod +x proc.com`;
close (PROC);
close (OUTPUT);
return;
}
#####
sub rewrite_proc_com1 {
    open (PROC, "proc.com") or die "\nCan not open proc.com.\n";
    if (-e "temp") {`rm ./temp`;}
    open (OUTPUT, ">temp");

    while ($line = <PROC>) {
        @int = split (" ", $line);
        if ($int[3] eq "EXT") {
            print OUTPUT "| nmrPipe -fn EXT -x1 $x1 -xn $xn -sw
\\n";
        }
        else {print OUTPUT $line;}
    }
    `mv temp proc.com`; `chmod +x proc.com`;
    close (PROC);
    close (OUTPUT);
    return;
}
#####
sub rewrite_proc_com2 {
    open (PROC, "proc.com") or die "\nCan not open proc.com.\n";
    if (-e "temp") {`rm ./temp`;}
    open (OUTPUT, ">temp");

    $k = 0;
    while ($line = <PROC>) {
        @int = split (" ", $line);

```

```

        if ($k == 0 and $int[3] eq "PS") {
            $int[5] = $int[5] + $p0;
            $int[7] = $int[7] + $p1;
            foreach $i (0..8) {print OUTPUT "$int[$i] ";}
            print OUTPUT " $int[9]\n";
            $k++;
        }
        else {print OUTPUT $line;}
    }
    `mv temp proc.com`; `chmod +x proc.com`;
    close (PROC);
    close (OUTPUT);
    return;
}
#####
sub rewrite_proc_com3 {
    open (PROC, "proc.com") or die "\nCan not open proc.com.\n";
    if (-e "temp") {rm ./temp;}
    open (OUTPUT, ">temp");

    while ($line = <PROC>) {
        @int = split (" ", $line);
        if ($int[3] eq "SP") {
            print OUTPUT "#";
            print OUTPUT $line;
        }
        else {print OUTPUT $line;}
    }
    `mv temp proc.com`; `chmod +x proc.com`;
    close (PROC);
    close (OUTPUT);
    return;
}
#####
sub rewrite_proc_com6 {
    open (PROC, "proc.com") or die "\nCan not open proc.com.\n";
    if (-e "temp") {rm ./temp;}
    open (OUTPUT, ">temp");

    while ($line = <PROC>) {
        @int = split (" ", $line);

```

```

    if ($int[0] eq "#") {
        print OUTPUT "| ";
        $i = 0;
        $a = 1;
        while ($i == 0) {
            if ($int[$a] eq "\\") {
                $i++;
                print OUTPUT "$int[$a]\n";
            }
            else {print OUTPUT "$int[$a] ";}
            $a++;
        }
    }
    else {print OUTPUT $line;}
}
`mv temp proc.com`; `chmod +x proc.com`;
close (PROC);
close (OUTPUT);
return;
}

print "\ncomb.com is generated corrected. Execute process2DSpe.pl.\n";

```

Appendix 6 process2DSpe.pl

```
#!/usr/bin/perl -w

#
# Run this script by typing the name of this script and press "Enter"
# It can pretty much guide you through the whole data analysis process.
#

print "\nRun this script by typing the name of this script and press ENTER.\n";
print "\nIt can pretty much guide you through the whole data analysis process.\n";
print "\nFunctions of this specific script:";
print "\n    1. This script will process a series of 2D spectra with different
        window function combinations controled by the following argument:\n";
print "    gg --> GM in both dimention\n";
print "    ll --> SP in both dimention\n";
print "    gl --> GM in H & SP in X(13C or 15N)\n";
print "    oo --> No window function in either dimension.\n";
print "\n    2. It will generate nmrview files.\n";
print "\nOf course, you can always modify this script according to your own
preferences.\n";

#
# Step 1.1, check the comb.com.
#
print "\nThe program will first call a subroutine \"check\" to check whether
comb.com is present.
It will also check whether both GM and SP are available for each dimention.
Proceed if yes, exit otherwise.\n";

print "\nYou can escape this step by inputing n and press ENTER, input y and
press ENTER otherwise.\n";
print "\nInput y and n for doing check and escaping checking, respectively.\n";
$arg = <STDIN>;
chomp ($arg);
if ($arg eq "y") {
    if (-e "comb.com" and !-z "comb.com") {check();}
    else {
        print "\ncomb.com dose not seem to exist.\n";
        print "\nThe script is terminated.\n";
        exit;
    }
}

```

```

    }
}
elseif ($arg eq "n") {
    print "\nSubroutine check() will not be called. Hence some functions of
this script may not work.\n";
    print "\nLet's go on.\n";
}
else {die "\nNo idea what you want.\n";}

print "\nStep one is finished. It's time for Step two, which is to generate
comb.com1 from comb.com
    and process all the spectra using comb.com1.\n";
print "\nLuckily enough, you can choose not to do step two. Anyhow, you have to
proceed.\n";
print "\nPress Enter to proceed.\n";
$arg = <STDIN>;
if ($arg eq "n") {print "Let's go on.\n";}

#
# Step 2, generating comb.com1 and process all the spectra
#
print "\nSTEP TWO, generating comb.com1 and process all the spectra.\n";
print "\nDo you want to generate appropriate a \"comb.com1\" and process the
spectra?\n";
print "\nYou can choose not to do so by input n for no, otherwise just input y for
yes and press ENTER\n";
$arg = <STDIN>; chomp ($arg);
if ($arg eq "y") {
    print "You will be asked to input the right window function combination
(gg/ll/gl/oo) after reading throught the following instructions.\n";
    print "You can only process the data using one window function
combination at a time for now.\n";
    print "No matter which combination you input (as long as it is right), a
subroutine will be called to generate
comb.com1 containing only that window function combination.\n";
    print "After comb.com1 is generated in this master directory, it will be
copied to each individual .fid directory and be executed.\n";
    print "This work is to be done by the subroutine process()";
    print "The input file of comb.com1 is fid and the output one is test.ft.\n";

    print "\nPress Enter to proceed.\n";
}
}

```

```

$arg = <STDIN>;
if ($arg eq "\n") {print "Let's go on.\n";}

comb_gg(); comb_ll(); comb_gl(); comb_oo();
print "Please input the right window function combination (gg/ll/gl/oo)\n";
$arg = <STDIN>;
chomp($arg);
if ($arg eq "gg") {
    $comb = "comb_gg.com";
    process();
}
elseif ($arg eq "ll") {
    $comb = "comb_ll.com";
    process();
}
elseif ($arg eq "gl") {
    $comb = "comb_gl.com";
    process();
}
elseif ($arg eq "oo") {
    $comb = "comb_oo.com";
    process();
}
else {
    print "\nI have no idea what do you want to do. Doublecheck your
input.\n";
    print "The script will be terminated.\n";
    exit;
}
}
elseif ($arg eq "n") {
    print "\ncomb.com1 will not be executed.\n";
}
else {
    print "\nI have no idea what you want.\n";
    print "\nThe script is terminated.\n";
    exit;
}

print "Now Step two is done or is skipped. It is time for step three.\n";
print "Step three is to generate nmrview files if you want.\n";

```

```

print "Again, you have a choice not to do so.\n";
print "\nPress Enter to proceed.\n";
$arg = <STDIN>;
if ($arg eq "\n") {print "Let's go on.\n";}

#
# Step 3, generate nmrview files if necessary.
#
print "\nSTEP THREE, generate nmrview files in the master directory/nvfiles.\n";
print "\nAgain you can choose not to do so by input n and press ENTER when
being asked to make the decision.\n";
print "\nDo you want to generate nmrview files?\n";
print "\nInput y/n for yes and no, respectively:\n";
$arg = <STDIN>;
chomp ($arg);
if ($arg eq "y") {
    print "A subroutine nvfile_generation() will be called to generate a file
called, convert.com, which take test.ft as inputfile to generate .nv files in
\"./nvfiles\".\n";
    nvfile_generation();
}
elseif ($arg eq "n") {print "\nnvfiles will not be generated.\n";}
else {
    print "\nI have no idea what you want.\n";
    print "\nThe script is terminated.\n";
    exit;
}

print "\nYou will run nmrview to extract intensity of each peak across all ncyc.\n";
print "\nGood luck!\n";

#
# Step 4, read .nv files into .nmrview
#
chdir "./nvfiles";
`ls ncyc_*.nv > temp1`;
$currentpath_nvfiles=`pwd`;
chomp($currentpath_nvfiles);
open(TEMP, ">temp_nmrview");
open(FILE, "temp1") or die "\nCan not open temp1.\n";
while ($line = <FILE>) {

```

```

    chomp($line);
    print TEMP "set fileName {";
    print TEMP $currentpath_nvfiles;
    print TEMP "\v";
    print TEMP $line;
    print TEMP "}\n";
    print TEMP "catch \"nv_dataset open {\$";
    print TEMP "fileName}\n";
}
close(TEMP);
close(FILE);
`mv temp_nmrview .nmrview`;
`rm temp1`;
#####
#####
sub check {
my $comb = "comb.com";
open (COMB, $comb);
my $i = 0;
my $j = 0;
while ($line = <COMB>) {
    if ($line ne "\n") {
        @int = split(" ", $line);
        if ($int[0] eq "|" and $int[3] eq "SP") {$i++;}
        if ($int[0] eq "|" and $int[3] eq "GM") {$j++;}
    }
}
if ($i != 2 or $j != 2) {
    print "\nI do not think both GM and SP are expected to be there available
for each dimation.\n";
    print "\nThe script will be terminated.\n";
    exit;
}
if ($i == 2 or $j == 2) {print "\ncomb.com exists and seems to be appropriate.\n";}
close (COMB);
return;
}
#####
sub check1 {
$comb = "comb.com1";
open(COMB, $comb);

```

```

$i = 0;
$j = 0;
while ($line = <COMB>) {
    @int = split(" ", $line);
    if ($int[0] eq "|" and $int[3] eq "SP") {$i++;}
    if ($int[0] eq "|" and $int[3] eq "GM") {$j++;}
}
}
#####
#####
sub comb_oo {
    print "Generate appropriate processing nmrPipe file.\n";
    if (-e "comb.com1") {`rm ./comb.com1`};
    $comb = "comb.com";
    open(COMB, $comb);
    open(OUTPUT, ">comb.com1");
    while ($line = <COMB>) {
        if ($line ne "\n") {
            @int = split(" ", $line);
            if ($int[3] ne "SP" and $int[3] ne "GM") {print OUTPUT $line;}
        }
    }
    `chmod +x comb.com1`;
    `mv comb.com1 comb_oo.com`;
    close (COMB);
    close (OUTPUT);
    return;
}
#####
#####
sub comb_gg {
    print "Generate appropriate processing nmrPipe file.\n";
    if (-e "comb.com1") {`rm ./comb.com1`};
    $comb = "comb.com";
    open(COMB, $comb);
    open(OUTPUT, ">comb.com1");
    while ($line = <COMB>) {
        if ($line ne "\n") {
            @int = split(" ", $line);
            if ($int[3] ne "SP") {print OUTPUT $line;}
        }
    }
}

```

```

}
`chmod +x comb.com1`;
`mv comb.com1 comb_gg.com`;
close (COMB);
close (OUTPUT);
return;
}
#####
sub comb_ll {
print "Generate appropriate processing nmrPipe file.\n";
if (-e "comb.com1") {`rm ./comb.com1`};
$comb = "comb.com";
open(COMB, $comb);
open(OUTPUT, ">comb.com1");
while ($line = <COMB>) {
    if ($line ne "\n") {
        @int = split(" ", $line);
        if ($int[3] ne "GM") {print OUTPUT $line;}
    }
}
}
`chmod +x comb.com1`;
`mv comb.com1 comb_ll.com`;
close (COMB);
close (OUTPUT);
return;
}
#####
sub comb_gl {
print "Generate appropriate processing nmrPipe file.\n";
if (-e "comb.com1") {`rm ./comb.com1`};
$comb = "comb.com";
open(COMB, $comb);
open(OUTPUT, ">comb.com1");
$i = 0;
$j = 0;
while ($line = <COMB>) {
    if ($line ne "\n") {
        @int = split(" ", $line);
        if ($int[3] ne "GM" and $int[3] ne "SP") {print OUTPUT $line;}
        if ($i == 0 and $int[3] eq "GM") {
            print OUTPUT $line;
        }
    }
}
}

```

```

                $i++;
            }
        }
        if ($j == 0 and $int[3] eq "TP") {$j++;}
        if ($j == 1 and $int[3] eq "SP") {print OUTPUT $line;}
    }
    `chmod +x comb.com1`;
    `mv comb.com1 comb_gl.com`;
    close (COMB);
    close (OUTPUT);
    return;
}
#####
sub process {
    $currentpath = `pwd`;
    chomp($currentpath);
    $currentpath =~ /(.)\./s;
    $parentpath = $1;
    $currentpath =~ /.+\./s;

    open (NAME, "exp_names") or die "\nCan not open exp_names.\n";

    print "\nProcessing Spectra\n\n";
    while ($line = <NAME>) {

        chomp ($line);
        $directory = $parentpath."/".$line;

        `cp $comb $directory`;
        print "\nexecuting comb.com1 in $directory\n";
        chdir $directory;
        `chmod +x $comb`;
        `./$comb`;
        if ($? != 0) {print "\nerror produced in executing comb.com1 in
$workingdirectory[8]\n";}
        else {print "\nno errors\n";}
    }
    chdir $currentpath;
}
return;
}

```

```
#####
#####
sub nvfile_generation {
    $currentpath = `pwd`;
    chomp($currentpath);
    $currentpath =~ /(.)\./+$/s;
    $parentpath = $1;
    $currentpath =~ /.+\/(.)$/s;
    $basename = $1;
    chomp($basename);
    $sarg = $parentpath."/".$basename."*";
    @spectra2proc = `ls -ald $sarg`;
    print "\nProcessing Spectra\n\n";

    chdir $currentpath;
    if (!-e "nvfiles") { `mkdir nvfiles`; }

    open (NCYC, "ncyccount") or die "\nCan not open ncyccount.\n";
    @line = <NCYC>;
    @int = split (" ", $line[0]);
    @int1 = split (" ", $line[1]);
    $i = 0;
    open (OUTPUT1, ">time.txt");

    foreach $spectrum (@spectra2proc) {
        @workingdirectory = split(" ", $spectrum);
        if ($workingdirectory[8] ne $currentpath) {
            chdir $workingdirectory[8];
            open (OUTPUT, ">convert.com");
            print OUTPUT '#!/bin/csh';
            print OUTPUT "\n\n";
            print OUTPUT "nrmPipe -in test.ft \\";
            print OUTPUT "\n";
            print OUTPUT "| pipe2xyz -nv -
out ../ncyc/nvfiles/ncyc_$i.nv\n";
            close (OUTPUT);
            print OUTPUT1 "$i $int[$i]\n";
            $i++;
            `chmod +x convert.com`;
            `./convert.com`;
        }
    }
}
```

```
        if ($? != 0) {print "\nerror produced in executing
convert.com in $workingdirectory[8]\n";}
        else {print "\nno errors\n";}
    }
}
close (NCYC);
close (OUTPUT1);
chdir $currentpath;
`mv time.txt ./nvfiles`;
return;
}
```

Appendix 7 R2calculation.pl

```
#!/usr/bin/perl -w

#
# This script will calculated R2 values from intensity file temp.
#

# To determine length of relaxtion time period
`cp ../../data.fid/procpar .`;
open (PARA, "procpar") or die "\nCan not open procpar.\n";

$j = 0;
$k = 0;
while ($line = <PARA>) {
    @int = split(" ", $line);
    if ($int[0] eq "time_T2") {foreach $i (1..10) {if ($int[$i] = ~^\d+/) {$j++;}}}
    if ($j == 10 and $int[0] == 1 and $k == 0) {
        $k++;
        $timeT2 = $int[1];
    }
}
close (PARA);
`rm procpar`;

print "\nThe input files are output files from nmrview/Rate Analysis.\n";
print "\nThe intensity file generated by the macro PrintRateData should be
assigned at this stage.\n";
print "\nIs the intensity file called temp?\n";
print "\nInput y and n for yes and no, respectively:\n";
$arg = <STDIN>;
chomp ($arg);
if ($arg eq "y") {
    $input = "temp";
    if (!-e $input) {die "\nI do not think $input existes.\n";}
}
elseif ($arg eq "n") {
    print "\nInput the name of the intensity file generate using the macro
PrintRateData.\n";
    $arg = <STDIN>;
    chomp ($arg);
```

```

    $input = $arg;
    if (!-e $input) {die "\nI do not think $input existes.\n";}
}
else {die "\nNo idea what you want.\n";}

# Step 1, judge how many ncyc_0 measured. Min = 1 and Max = 2
if (!-e "ncyccount") {print "\nPlease copy ncyccount in this directory.\n";}
`cp ../ncyccount ./`;

zero();

# Step 2, calculate R2 apparent based on the intensity file
r2_calculation();

# Step 3, find out the duplicated data point and calculate STD
std_calculation();
`rm r2app_nv`;
#####
sub std_calculation {

open (INPUT, "r2app_nv") or die "\nCan not open r2app_nv";
@line = <INPUT>;
@int = split (" ", $line[0]);
$num_dup = 0;
@duplicate1 = ();
@duplicate2 = ();
foreach $i (1..$count-$zero) {
    my $j = 2*$i - 1;
    my $k = 2*$i + 1;
    if ($int[$j] == $int[$k]) {
        $duplicate1[$num_dup] = 2*$i;
        $duplicate2[$num_dup] = 2*$i + 2;
        $num_dup++;
        print "\nGood\n";
    }
}
if ($num_dup != $dup) {die "\nNo. of duplicate is not correct.\n";}
close (INPUT);

open (INPUT, "r2app_nv") or die "\nCan not open r2app_nv";
if (-e "R2app") {`rm ./R2app`;}

```

```

open (OUTPUT, ">R2app");
while ($line = <INPUT>) {
    @int = split (" ", $line);
    chomp ($line);
    print OUTPUT $line;

    $std_ave = 0;
    foreach $i (0..$num_dup-1) {
        $a = $int[$duplicate1[$i]]; $b = $int[$duplicate2[$i]];
        std();
        $std_ave = $std_ave + $std;
        print $std_ave;
        print "\nGreat\n";
    }

    $std_ave = $std_ave / $num_dup;

    print OUTPUT " $std_ave\n";
}
close (INPUT);
close (OUTPUT);
return;
}
#####
sub std {
    $ave = ($a + $b)/2;
    $std = sqrt(($a - $ave)**2 + ($b - $ave)**2);
    return;
}
#####
sub r2_calculation {

open (INPUT, $input) or die "\nCan not open $input.\n";
if (-e "r2app_nv") {`rm ./r2app_nv`;}
open (OUTPUT, ">r2app_nv");
while ($line = <INPUT>) {
    @int = split (" ", $line);
    $negativeness=0;
    foreach $i (1..$count){
        if ($int[2*$i] < 0){
            $negativeness++;

```

```

    }
  }
  if($negativeness==0){
    if ($zero == 1) {
      if ($int[1] != 0) {
        print "\nThere is not ncyc_0 at all.\n";
        exit;
      }
      if ($int[1] == 0 and $int[3] > 0) {print "\nAll right.\n";}
      if ($int[1] == 0 and $int[3] == 0) {die "No. of ncyc_0 is not
right.\n";}
    }
    if ($zero == 2) {
      if ($int[1] != 0) {
        print "\nThere is not ncyc_0 at all.\n";
        exit;
      }
      if ($int[1] == 0 and $int[3] > 0) {die "\nonly one ncyc_0.\n";}
      if ($int[1] == 0 and $int[3] == 0 and $int[5] > 0) {print "\nAll
right.\n";}
    }
  }

  if ($int[2*$count] ne "" and $int[2*$count+1] eq "") {print "\nAll
right.\n";}
  else {die "No. of data is not right.\n";}

  print "\nLet's start calculating r2app_nv.\n";

  print OUTPUT $int[0];

  if ($timeT2==0.04) {
    if ($zero == 1) {
      foreach $i (2..$count) {
        $j = 2*$i;
        $step = $int[$j-1]/40;
        print OUTPUT " $step";
        $r2 = (-25)*log( $int[$j] / $int[2] );
        print OUTPUT " $r2";
      }
      print OUTPUT "\n";
    }
  }

```

```

        if ($zero == 2) {
            foreach $i (3..$count) {
                $j = 2*$i;
                $step = $int[$j-1]/40;
                print OUTPUT " $step";
                $r2 = (-25)*log( 2*$int[$j] / ( $int[2] + $int[4] ) );
                print OUTPUT " $r2";
            }
            print OUTPUT "\n";
        }
    }
elseif ($timeT2==0.02){
    if ($zero == 1) {
        foreach $i (2..$count) {
            $j = 2*$i;
            $step = $int[$j-1]/20;
            print OUTPUT " $step";
            $r2 = (-50)*log( $int[$j] / $int[2] );
            print OUTPUT " $r2";
        }
        print OUTPUT "\n";
    }
    if ($zero == 2) {
        foreach $i (3..$count) {
            $j = 2*$i;
            $step = $int[$j-1]/20;
            print OUTPUT " $step";
            $r2 = (-50)*log( 2*$int[$j] / ( $int[2] + $int[4] ) );
            print OUTPUT " $r2";
        }
        print OUTPUT "\n";
    }
}
else{print "\nSomething is wrong with time_T2.\n";}
}
}
}
close (INPUT);
close (OUTPUT);
return;
}
#####

```

```

sub zero {

open (NCYC, "ncyccount") or die "\nCan not open ncyccount.\n";
while ($line = <NCYC>) {
    @int = split (" ", $line);

    # Determined how many experiments in the parent directory.
    print "\nDetermined how many experiments in the parent directory.
Assuming there is less or equal to 30 experiments.\n";
    $count = 0;
    foreach $i (0..30) {if ($int[$i] =~ ^d+/) {$count++;}}
    print "\nThe number of experiments is $count\n";

    # Determine how many nyc_0 are done, minimum is 1 and maximum is 2.
    print "\nDetermine how many nyc_0 are done, minimum is 1 and
maximum is 2.\n";
    $zero = 0;
    foreach $i (0..$count-1) {if ($int[$i] == 0) {$zero++;}}

    print "\nDetermine duplicate experiments except nyc_0.\n";
    $dup = 0;
    foreach $i ($zero..$count-2) {
        $a = $i + 1;
        foreach $j ($a..$count-1) {if ($int[$i] == $int[$j]) {$dup++;}}
    }
    print "\nThere are $dup duplicates besides nyc_0.\n";

}
close (NCYC);
return;
}

```

Appendix 8 inputgeneration.pl

```
#!/usr/bin/perl -w

print "\n      This script will take R2app and generate input files for simulated-
annealing added Dmitry's program\n";
print "\n      Make sure this script is executed in the same folder where the data
file to be changed is in.\n";
print "\n      The default model index is 3 and will be changed accordingly during
cpmgdata generation.\n";
print "\n      For one line of data,
      model index      (31, 51, 71 for 2-, 3- and 4-state equilibrium)
      constant time index (1 and 2 for 20ms and 40ms)
      temperature index (1, 2, and 3 for 5degree, 10degree, and 15
degree)
      static field index (1, 2, and 3 for 600MHz, 800MHz, and
900MHz)
      protein index     (1, 2, 3, 4, and 5 for ADwt, ADK208V,
ADK208A, ADK208E, and DH)
      residue index     (refer to Appendix)
      ncy              (2 to 40)
      static field strength in radian/second
      tau              in s
      R2              in 1/s
      deltaR2         in 1/s\n";

print "\nPlease input the name of the file to be changed:\n";
$arg=<STDIN>;
chomp($arg);
$input=$arg;

print "\nInput the duration of constant relaxation period in ms, 20 or 40:\n";
$arg=<STDIN>;
chomp($arg);
$constantT=$arg;
if ($constantT == 20 or $constantT == 40) {};
else{
    print "\nThe constant time is neither 20 ms nor 40 ms. WRONG input.\n";
    print "\nInput the duration of constant relaxation period in ms, 20 or 40:\n";
    $arg=<STDIN>;
    chomp($arg);
```

```

    $constantT=$arg;
}

print "\nInput the magnetic field in MHz, 600, 800 or 900:\n";
$arg=<STDIN>;
chomp($arg);
$staticField=$arg;
if ($staticField == 600 or $staticField == 800 or $staticField == 900) {};
else{
    print "\nThe static field is neither 600 MHz nor 800 MHz nor 900 MHz.
    WRONG input.\n";
    print "\nInput the magnetic field in MHz, 600, 800 or 900:\n";
    $arg=<STDIN>;
    chomp($arg);
    $staticField=$arg;
}

print "\nInput temperature index:\nMake sure no mistake here!!!!!!!!!!\n";
print "5degree is 1, 10degree is 2 and 15 degree is 3\n";
$arg=<STDIN>;
chomp($arg);
$temperatureIn=$arg;
$int1[2]=$temperatureIn;

print "\nInput the protein index:\nMake sure no mistake here!!!!!!!!!!\n";
$arg=<STDIN>;
chomp($arg);
$proteinIn=$arg;

$int1[0] = 71;
if ($constantT == 20) {$int1[1] = 2;}
else {$int1[1] = 1;}

if ($staticField == 600) {$int1[7] = 946.94;$int1[3] = 1;}
elsif ($staticField == 800) {$int1[7] = 1264.7;$int1[3] = 2;}
elsif ($staticField == 900) {$int1[7] = 1421.4;$int1[3] = 3;}
else {die "\nWrong field.\n";}

$int1[4] = $proteinIn;

$inputflag = 'y';

```

```

while ($inputflag eq 'y'){

    print "\nInput the residue index:\nMake sure no mistake here!!!!!!!!!!\n";
    $arg=<STDIN>;
    chomp($arg);
    $residueIn=$arg;

    print "\nInput the resonance number on the spectrum:\nMake sure no
mistake here!!!!!!!!!!\n";
    $arg=<STDIN>;
    chomp($arg);
    $peakIn=$arg;

    $resonance = 'ncyc_0.';
    $resonance.=$peakIn;

    $int1[5] = $residueIn;

#    if ($residueIn%2==1) {print "\nThis is a residue assigned as a front
dynamic residues.\n";}
#    else {print "\nThis is a residue assigned as a back dynamic residues.\n";}

    # Try to determine whether there is a mistake when calculating input file
for palmer program.
    open(INPUT, $input);
    $mcpmg = 0;
    @line=<INPUT>;
    @int = split(" ", $line[0]);
    my $size = @int;
    $mcpmg=$int[$size-3];
    close(INPUT);

    if ($mcpmg == 0.5) {die "\nThere is a mistake when generating palmer
input. Double-check before persuing further.\n"}
    if ($mcpmg == 1.0) {$factor = 40;}

    if(-e "temp_cpmggeneration"){`rm temp_cpmggeneration`;}
    open (OUTPUT, ">temp_cpmggeneration");
    open(INPUT, $input);

    while ($line = <INPUT>) {

```

```

    @int = split(" ", $line);
    if ($int[0] eq $resonance) {
        for $j (1..$size/2-1){
            $int1[6] = $factor*$int[2*$j-1];
            $int1[8] = 0.04/($int1[6]*4);
            $int1[9] = $int[2*$j];
            $int1[10] = $int[$size-1];

            for $i (0..10) {print OUTPUT "$int1[$i] ";}
            print OUTPUT "\n";
        }
    }
}

close (INPUT);
close (OUTPUT);

if ($constantT == 20) {
    if(-e "dynamic20"){
        `cat dynamic20 temp_cpmggeneration > temp_dynamic20`;
        `mv temp_dynamic20 dynamic20`;
    }
    else {`mv temp_cpmggeneration dynamic20`;}
}
elseif ($constantT == 40) {
    if(-e "dynamic40"){
        `cat dynamic40 temp_cpmggeneration > temp_dynamic40`;
        `mv temp_dynamic40 dynamic40`;
    }
    else {`mv temp_cpmggeneration dynamic40`;}
}

print "\nDo you have more peaks to change?\nInput y for yes and n for no
and the press ENTER:";
$arg=<STDIN>;
chomp($arg);
$inputflag=$arg;
}
`rm temp_cpmggeneration`;

```

Appendix 9 generatecpmgdata.pl

```
#!/usr/bin/perl -w

# This script will generate data file, cpmgdata.
print "\nThis script will generate data file, cpmgdata.\n";

# Delete cpmgdata if it exists.
print "\nDelete cpmgdata if it exists.\n";
if (-e "cpmgdata") {`rm cpmgdata`;}

# Open cpmgdata for input.
print "\nOpen cpmgdata for input.\n";
open OUTPUTDATA, ">cpmgdata";

print "\nGenerate cpmgdata as you wish.\n";
print "\nFirst of all, you should specify which model you are wishing to use.\n";
print "\nThe following model index are used:
      2  2-site CR model
     31  2-site model
     51  General 3-site model
     71  General 4-site model
\n";
print "\nWhich model???\n";
$arg = <STDIN>;
chomp ($arg);
$modelIndex = $arg;

print "\nSecond of all, you should specify how much noise you want for 40 ms
data in percentage:\n";
$arg = <STDIN>;
chomp ($arg);
$noise40 = $arg/100.0;

print "\nThird of all, you should specify how much noise you want for 20 ms data
in percentage:\n";
print "\nSpecify even if you are not going to use 20 ms data since it will not used
and will not affect end results\n";
$arg = <STDIN>;
chomp ($arg);
$noise20 = $arg/100.0;
```

```

print "\nYou can specify which peaks you want to use.\n";
print "\nDo you want to specify which peaks to use?\n";
print "\nInput y and press ENTER for yes, otherwise input n and press
ENTER:\n";
$arg = <STDIN>;
chomp ($arg);

$notemperature=0; # Indexing how many temperatures used (Normally 3,
5degree, 10degree, and 15degree)
$nopeak=0; # Indexing how many peaks used (<=31).
$nofield=0; # Indexing how many fields used (Normally 3, 600 MHz,
800 MHz, and 900 MHz).
$noprotein=0; # Indexing how proteins used (up to 5).
$noexperiments=1; # Index how many relaxation period used (One if only 40
ms used and two if 20 ms also used).

if ($arg eq "y") {
  print "\nInput peak index to be used and input -1 if done:\n";
  print "\nMake sure one index only input once since this program doesn't
doublecheck\n";
  print "\nYou can doublecheck before input -1 since all inputs are still on the
screen.\n";
  $arg1 = <STDIN>;
  chomp ($arg1);
  while ($arg1 != -1){
    $peakindexes[$nopeak]=$arg1;
    print "\n$nopeak, $peakindexes[$nopeak]\n";
    open (INPUT40, "dynamic40") or die "\nCan not open dynamic40.\n";
    $i = 0;
    while ($line = <INPUT40>) {
      @int = split (" ", $line);
      if ($int[2] > $notemperature) {$notemperature = $int[2];}
      if ($int[4] > $noprotein) {$noprotein = $int[4];}
      if ($int[3] > $nofield) {$nofield = $int[3];}
      if ($int[5] == $arg1 and $int[9]<60.0) {
        $noise = $noise40*$int[9];
        if($noise < $int[10]) {$noise = $int[10];}
        print OUTPUTDATA "$modelIndex $int[1] $int[2] $int[3] $int[4] $int[5]
$int[6] $int[7] $int[8] $int[9] $noise\n";
        $i++;
      }
    }
  }
}

```

```

    }
  }
  close (INPUT40);

  if ($i == 0){
    print "\nThere are not data for peak $arg1.\n";
  }
  else {print "\nThere $i data point for peak $arg1.\n";}
#       print "\nDo you want to use 20 ms data for peak No. $arg1 ?\n";
#       print "\nInput y and press ENTER for yes, otherwise input n and
press ENTER:\n";
#       $arg2 = <STDIN>;
#       chomp ($arg2);
#       if ($arg2 eq "y") {
#           $noexperiments=2;
#           open (INPUT20, "dynamic20") or die "\nCan not open
dynamic20.\n";
#           while ($line = <INPUT20>) {
#               @int = split (" ", $line);
#               if ($int[4] > $noprotein) {$noprotein = $int[4];}
#               if ($int[9] > $nofield) {$nofield = $int[9];}
#               if ($int[5] == $arg1) {
#                   $noise = $noise20*$int[7];
#                   print OUTPUTDATA "$modelIndex 2 $int[2] $int[3]
$int[4] $int[5] $int[6] $int[7] $noise $int[9]\n";
#                   }
#               }
#           close (INPUT20);
#       }
#       elsif ($arg2 eq "n") {
#           print "\nWell then, only 40 ms data are used.\n";
#       }
#       else {die "\nGame over. Restart the program if you want to do
more.\n";}

  print "\nInput peak index to be used and input -1 if done:";
  print "\nMake sure one index only input once since this program doesn't
doublecheck";
  print "\nYou can doublecheck before input -1 since all inputs are still on the
screen.\n";

```

```
$arg1 = <STDIN>;
chomp ($arg1);
$nopeak++;
}
}
elseif ($arg eq "n") {}
else {die "\nGame over. Restart the program if you want to do more.\n";}
close (OUTPUTDATA);

for $i (0..$nopeak-1) {print "\n$i, $peakindexes[$i]\n";}

print "\nThe data file cpmgdata is generated.\n";
print "\nThis is the end.\n";
```

Appendix 10 curvefitting.pl

```
#!/usr/bin/perl -w

# This script will generate input files, kfile, pfile, wfile, rfile.
# It can modify c code and compile it before running if needed.
# In the last, it will initiate cpmgfit and store final results in file, results.

print "\n      This script will generate input files, kfile, pfile, wfile, rfile.\n";
print "\n      It can modify c code and compile it if needed.\n";
print "\n      In the last, it will initiate cpmgfit and store final results in file,
results.\n";
print "\n      The results will be saved in the same directory, too.\n";

print "\nDelete kfile, pfile, wfile, and rfile, if they exist.\n";
if (-e "kfile") {`rm kfile`;}
if (-e "pfile") {`rm pfile`;}
if (-e "wfile") {`rm wfile`;}
if (-e "rfile") {`rm rfile`;}

print "\nOpen kfile, pfile, wfile, rfile for input.\n";
open OUTPUTK, ">kfile";
open OUTPUTP, ">pfile";
open OUTPUTW, ">wfile";
open OUTPUTR, ">rfile";

print "\nFirst of all, you should specify which model in use.\n";
print "\nThe following model index are used:
      2  2-site model CR
      31 2-site model
      51 General 3-site model
      71 General 4-site model
\n";
print "\nWhat is the model index????\n";
$arg = <STDIN>;
chomp ($arg);
$modelIndex = $arg;

print "\nGenerate kfile assuming five proteins 30 = 5x6.\n";
for $i(1..30){
```

```

# each temperature has two lines: one for time_T2 = 20 ms, another for 40 ms
# low and high boundaries of kex1, kex2, kex3, and kex4 will be filled in c code
# If there are not data for protein 5, the corresponding kex2 is set to known
print OUTPUTK "7.0 0.0 7.0 0.0 7.0 0.0 7.0 0.0 f f f f\n";
}
close (OUTPUTK);

print "\nGenerate pfile assuming five proteins 30 = 5x6.\n";
for $i(1..25){
  # each temperature has two lines: one for time_T2 = 20 ms, another for 40 ms
  # low and high boundaries of pAC, pAO, pBO, pBC will be filled in c code
  print OUTPUTP "0.25 0.0 0.25 0.0 0.25 0.0 0.25 0.0 f f f f\n";
}
for $i(1..5){
  # each temperature has two lines: one for time_T2 = 20 ms, another for 40 ms
  # low and high boundaries of pAC, pAO, pBO, pBC will be filled in c code
  print OUTPUTP "0.0 0.0 0.95 0.0 0.05 0.0 0.0 0.0 f f f f\n";
}
close (OUTPUTP);

print "\nGenerate rfile assuming 31 resonances 186 = 31 * 2 * 3 temperatures * 5
proteins.\n";
for $i (1..930){
  # low and high boundaries of 600 R2c R2o, 800 R2c R2o, 900 R2c R2o
  # each temperature has two lines: one for time_T2 = 20 ms, another for 40 ms
  print OUTPUTR "\ 25.0 1.0 25.0 1.0 25.0 1.0 25.0 1.0 25.0 1.0 25.0 1.0 f f f f f
f\n";
}
close (OUTPUTR);

print "\nGenerate wfile assuming 62= 31 * 2.\n";
for $i (1..62){
  # each temperature has two lines: one for time_T2 = 20 ms, another for 40 ms
  # low and high boundaries of WAC WAO WBO WBC in which WAC is fixed
  at zero
  print OUTPUTW "\ 0.0 .0 0.0 .0 0.0 .0 0.0 .0 f f f f\n";
}
close (OUTPUTW);

open (DATA, "cpmgdata") or die "\nCan not open cpmgdata.\n";

```

```

# go line by line and change whatever need to be changed in kfile, pfile, wfile and
rfile.
# We assume DH data are included in analysis for every temperature and every
field
# where at least one of protein 1-4 is included.
while ($line = <DATA>) {
    @int = split (" ", $line);

    # rewrite kfile
    open(KFILE, "kfile") or die "\nCan not open kfile.\n";
    open(OUTPUTK, ">kfile_temp");
    $i = 0;
    while ($linek=<KFILE>) {
        $i++;
        if ($int[4]*6-6+$int[2]*2-2+$int[1] != $i) {
            print OUTPUTK $linek;
        }
        else {
            if ($int[4] == 5) {
                # kex2 corresponds to the open edge kex
                print OUTPUTK "7.0 0.0 7.0 0.0 7.0 0.0 7.0 0.0 f u f fn";
            }
            elsif ($int[4] == 1) {
                # kex2 of protein1-4 = kex of protein 5
                # kex4 assumed identical for protein 1-4.
                print OUTPUTK "7.0 0.0 7.0 0.0 7.0 0.0 7.0 0.0 u f u \n";
            }
            else {
                print OUTPUTK "7.0 0.0 7.0 0.0 7.0 0.0 7.0 0.0 u f u fn";
            }
        }
    }
}
close(KFILE);
close(OUTPUTK);
`mv kfile_temp kfile`;

# rewrite pfile
open(PFILE, "pfile") or die "\nCan not open pfile.\n";
open(OUTPUTP, ">pfile_temp");
$i = 0;
while ($linep=<PFILE>) {

```

```

$ i++;
if ($int[4]*6-6+$int[2]*2-2+$int[1] != $i) {
    print OUTPUTP $linep;
}
else {
    if ($int[4] == 5) {
        # pAO is adjustable. pAC and pBC are extremely small. pBO=1-pAO-
delta.
        print OUTPUTP "0.00001 0.0 0.50 0.0 0.99 0.0 0.00001 0.0 f u f f\n";
    }
    elsif ($int[4] == 1) {
        # pAC and pBC adjustable. pAO and pBO calculated.
        print OUTPUTP "0.4 0.0 0.99 0.0 0.99 0.0 0.4 0.0 u f f u\n";
    }
    else {
        # pAC adjustable. pAO, pBO and pBC calculated. for protein 2, 3, 4
        print OUTPUTP "0.4 0.0 0.99 0.0 0.99 0.0 0.4 0.0 u f f f\n";
    }
}
}
}
close(PFILE);
close(OUTPUTP);
`mv pfile_temp pfile`;

# rewrite rfile
open(RFILE, "rfile") or die "\nCan not open rfile.\n";
open(OUTPUTR, ">rfile_temp");
$i = 0;
while ($liner = <RFILE>) {
    $i++;
    if ($int[4]*186-186+$int[5]*6-6+$int[2]*2-2+$int[1] != $i) {
        print OUTPUTR $liner;
    }
    else {
        @intr=split(" ", $liner);
        if ($int[3] == 1){
            $intr[12]='u';
            $intr[13]='f';
            if($int[6]==40){$intr[0]=$int[9];}
            if($int[5]<8){
                $intr[2]=5.0;
            }
        }
    }
}

```

```

    }
}
elseif ($int[3] == 2){
    $intr[14]='u';
    $intr[15]='f';
    if($int[6]==40){$intr[4]=$int[9];}
    if($int[5]<8){
        $intr[6]=5.0;
    }
}
elseif ($int[3] == 3){
    $intr[16]='u';
    $intr[17]='f';
    if($int[6]==40){$intr[8]=$int[9];}
    if($int[5]<8){
        $intr[10]=5.0;
    }
}
else {die "\nSomething wrong with static field index.\n";}

print OUTPUTR " ";
for $j (0..16) {
    print OUTPUTR $intr[$j];
    print OUTPUTR " ";
}
print OUTPUTR "$intr[17]\n";
}
}
close(RFILE);
close(OUTPUTR);
`mv rfile_temp rfile`;

# rewrite wfile
open(WFILE, "wfile") or die "\nCan not open wfile.\n";
open(OUTPUTW, ">wfile_temp");
$i=0;
while ($linew = <WFILE>) {
    $i++;
    if ($int[5]*2-2+$int[1] != $i) {
        print OUTPUTW $linew;
    }
}

```

```

else{
  # later more strengent constraint will be applied.
  if ($int[5] == 1 or $int[5] == 2 or $int[5] == 3 or $int[5] == 4 or $int[5] == 5
or $int[5] == 6 or $int[5] == 7 or $int[5] == 8 or $int[5] == 9 or $int[5] == 10 or
$int[5] == 25 or $int[5] == 26){
    print OUTPUTW "\ 0.6 0.0 0.0 0.0 0.0 0.0 0.0 0.6 0.0 u f f \n";
  }
  else {
    print OUTPUTW "\ 0.0 0.0 0.0 0.0 0.0 0.6 0.0 0.6 0.0 f f u f \n";
  }
}
}
close(WFILE);
close(OUTPUTW);
`mv wfile_temp wfile`;
}

```

```
exit;
```

```

print "\nDelete run.txt if it exists.\n";
if (-e "run.txt") {`rm run.txt`;}
print "\nOpen run.txt for writing.\n";
open OUTPUT, ">run.txt";
print OUTPUT "read cpmgdata d\n";
print OUTPUT "read kfile k\n";
print OUTPUT "read pfile p\n";
print OUTPUT "read wfile w\n";
print OUTPUT "read rfile R\n";
print OUTPUT "min results\n";
close (OUTPUT);

```

```

print "\nRun cpmgfit according to generated data file and input files.\n";
`./cpmg_fit_linux8 < run.txt`;

```

```
print "\nStore all results into one file allresults.\n";
```

```

#`cat cpmgdata kfile pfile wfile rfile results > results_temp`;
if(!-e "allresults") {`cat cpmgdata kfile pfile wfile rfile results > allresults`;}
else {`cat allresults results > allresults_temp`;`mv allresults_temp allresults`;}

```

```
print "\nThis is the end.\n";
```

Appendix 11 C code for curve fitting

```

#include <time.h>
#include <stdio.h>
#include <math.h>
#include <string.h>
#include <malloc.h>
#include <values.h>

#define VERSION 1.0

#define LINUX

#ifdef LINUX
#define _file _fileno
#define LN_MAXDOUBLE MAXDOUBLE
#define LN_MAXINT MAXINT
#endif

#ifdef MAIN
#define EXT
#else
#define EXT extern
#endif

#define FALSE 0
#define TRUE 1

/* Maximal Array Sizes */
#define MAXSTR 512 /* max string lenth */
#define MAXDATA 50000 /* max number data points (total) */
#define MAXPAR 2000 /* max number of adjustable parameters */

/* Some constants and defaults */
#define pi M_PI /* 3.1415926 */
#define MAXALLOC_RETR 5 /* max. # of retries in memory allocatios (rmalloc)
*/
#define H1_GM_RAT (2.6753e4) /* H1 hyromagnetic ratio */
#define N15_GM_RAT (-2.71e3) /* N15 hyromagnetic ratio */
#define C13_GM_RAT (6.728e3) /* C13 hyromagnetic ratio */
#define P31_GM_RAT (1.0785e4) /* P31 hyromagnetic ratio */
#define KB 1.38065812e-23 /* Boltzmann Kb */

```

```

#define HPL 6.626975540e-34 /* Planck h constant */
#define KB_HPL 2.0833910004140440814e+10 /* KB/HPL */
#define UNR 8.31451070 /* Universal gas constant R */
#define ABSZ 273.15 /* Absolute zero */

#define DTT2 0.04 /* default time_T2 */

#define EPSFCN_DEF 1.0e-08
#define R_COEFF 1e+6
#define P_COEFF 1e+8
#define K_COEFF 1e+6

/* nuclei */
#define NONE -1
#define H1 1 /* H1 */
#define N15 2 /* N15 */
#define C13 3 /* C13 */
#define P31 4 /* P31 */

/* coherences */
#define SQ 1 /* single quantum */
#define ZQ 2 /* zero quantum */
#define DQ 3 /* double quantum */
#define MQ 4 /* multiple quantum (cpmg on first nucleus starting from 50/50
dq/zq and
a single 180 pulse on a second nucleus */
#define RF 5 /* Rotating frame relaxation, R1r */

/* models */
#define SI 1 /* simple; assuming no exchange */
#define CR 2 /* CPMG data - Carver-Richards 2-site model (or modified CR for
MQ)*/
#define TS 3 /* 2-site model for comparison with CR */
#define GM 4 /* Genearl 3-site model, linear */
#define GN 5 /* General 3-site model, circle */
#define FM 6 /* General 4-site model, linear */
#define FN 7 /* General 4-site model, circle */

/* Defaults */
#define d_prompt "> "

```

```

/* Keywords */

#define N_WORDS 11 /* Number of keywords */

#define EXIT 1 /* Exiting from programm */
#define HELP 2 /* - */
#define SET 3
#define READ 4
#define DEL 5
#define WRITE 6
#define MIN 7
#define BACKUP 8
#define RESTORE 9
#define RUN 10

/* Variables, Data and Parameter structures */

EXT volatile int abrflg; /* abort flag */

EXT char cpmg_fit_dir[MAXSTR]; /* cpmg_fit directory */
EXT char prompt[MAXSTR]; /* cpmg_fit directory */

EXT struct dict
{
char w[10];
int iw; } words[N_WORDS];

EXT struct data
{
int coh_; /* coherence */
int mol_; /* model index */
int nucl_; /* nucleus */
int nucl1_; /* additional nucleus for CPMG data for dq,zq,mq coherences */
double xhgmrat_; /* ratio dfrq/hfrq for main nucleus */
double xhgmrat1_; /* ratio dfrq/hfrq for additional nucleus*/
double hsfreq_; /* 1H spectrometr frequency */
double temp_; /* temperature */
double time_T2_; /* total length of CPMG periods, for CPMG data only */
double cprfq_; /* CPMG freq 1/(4*d); tau=time_T2/(4*ncyc); or B1 field in
frequency units for R1r data */
double off_; /* offset from the carrier, for R1r data only */

```

```

int itdv_; /* element index in tdv array */
int ifdv_; /* element index in fdv array */
int ipdv_; /* element index in pdv array */

double *R1_; /* Pointer to the array R1_[3] with R1A, R1B and R1C, for R1r
data only */ /* Pointer to the array R1_[4] with R1A, R1B, R1C and R1D, for
R1r data only */
double *R2_; /* Pointer to the array R2_[3] with R2A, R2B and R2C */ /*
Pointer to the array R2_[4] with R2A, R2B, R2C and R2D*/
double *W_; /* Pointer to the array W_[3] with WA, WB and WC for main
nucleus */ /* Pointer to the array W_[4] with WA, WB, WC and WD for main
nucleus */
double *W1_; /* Pointer to the array W_[3] with WA, WB and WC for
additional nucleus */ /* Pointer to the array W1_[4] with W1A, W1B, W1C and
W1D for additional nucleus */
double *P_; /* Pointer to the array P_[3] with PA, PB and PC */ /* Pointer to
the array P_[4] with PA, PB, PC and PD */
double *KEX_; /* Pointer to the array KEX_[3] with KEX_AB, KEX_AC and
KEX_BC */ /* Pointer to the array KEX_[4] with KEX1, KEX2, KEX3, and
KEX4*/

int nr_; /*number of residues*/
int np_; /*number of protein*/
int nf_; /*number of static fields*/
int nd_; /*number of data point*/

int ncy_ ; /*number of d-180-d-d-180-d elements during time_T2*/
int proteinIn_; /*Index of protein, which determines which elements of kmatrix
and pmatrix to use*/
int residueIn_; /*Index of residue, which determines which elements of R2matrix
and dwmatrix to use*/
double val_; /*experimental R2 for residue "residueIn" of protein "proteinIn" at
static field "staField" and ncy_ "ncy_"/>
double err_; /*error in experimental R2*/
double residual_; /*error in experimental R2*/
double staField_; /* static field scaling factor which times ppm gives rad/s*/
int staFieldIn_; /* static field ID */
double tau_; /*duration of d in seconds*/
int contimIn_; /* Indexing the constant time 1 for 40 ms and 2 for 20 ms */
int temperatureIn_; /* Indexing temperature, 1, 2 and 3 for 5, 10, and 15 degree */

```

```

double cval_; /*calculated value of R2*/
double F_; /*target function*/
}dat[MAXDATA];

/* Minimization related general parameters */
EXT struct param
{
int mod_; /* calculation model; 2 - CR (Carver-Richards, CPMG; Trott/Palmer
R1r) 3 - GM (General 3-site model), 4, 5,6,7*/
int sim_; /* TRUE - simultaneous 180 pulse for MQ coherences at the middle
of relaxation period */
int min_; /* flag indicating whether minimization is done */
int clc_; /* flag indicating whether minimization is succesful */
int clcu_; /* flag indicating whether uncertaintie are sucessfully calculated */
double F_; /* target function over all data */
double P_lvl_; /* chi2 probability to get loss F > f by chance */
int df_; /* degrees of freedom */

int nnucl_; /* number nuclei types */
int ncoh_; /* number coherences types */
int ntemp_; /* number of temperatures */

int nke_; /* Number of kex read */
int npo_; /* Number of populations read */
int nr2_; /* Number of r2 read */
int nw_; /* Number of chemical shift read*/

int ntdv_; /* number of elements in structure tdv */
int nrdv_; /* number of elements in structure rdv */
int nfdv_; /* number of elements in structure fdv */
int npdv_; /* number of elements in structure pdv */

int res_[MAXPAR]; /* residues */
int nucl_[MAXPAR]; /* nuclei types */
int coh_[MAXPAR]; /* coherences */
int prot_[MAXPAR]; /* protein */
double temp_[MAXPAR]; /* temperatures */
double hsfrq_[MAXPAR]; /* fields */

```

```
char rat_mod_; /* model for temp. dependence of rate constants; 'n' - no model 'a'
- arrenius model */
```

```
/* parameter for arrenius model for rate and equilibrium constants */
```

```
double dHBA_, dSBA_; /* KeqBA */
```

```
double dHCA_, dSCA_; /* KeqCA */
```

```
double dH1BA_, dS1BA_; /* KBA */
```

```
double dH1CA_, dS1CA_; /* KCA */
```

```
double dH1CB_, dS1CB_; /* KCB */
```

```
double edHBA_, edSBA_; /* errors in KeqBA */
```

```
double edHCA_, edSCA_; /* errors in KeqCA */
```

```
double edH1BA_, edS1BA_; /* errors in KBA */
```

```
double edH1CA_, edS1CA_; /* errors in KCA */
```

```
double edH1CB_, edS1CB_; /* errors in KCB */
```

```
char fix_dHBA_, fix_dSBA_; /* fix flag for KeqBA */
```

```
char fix_dHCA_, fix_dSCA_; /* fix flag for KeqCA */
```

```
char fix_dH1BA_, fix_dS1BA_; /* fix flag for KBA */
```

```
char fix_dH1CA_, fix_dS1CA_; /* fix flag for KCA */
```

```
char fix_dH1CB_, fix_dS1CB_; /* fix flag for KCB */
```

```
double cov_hs_[10][10]; /* covariance matrix for H and S variables */
```

```
double p_lvl_; /* chi2 probability to get loss F>f by change */
```

```
int ndata_; /* number of data point */
```

```
int npar_k_; /* number of elements in kmatrix, for 2-, 3-, and 4-sites,
there are 1, 3, 4 elements in kmatrix per protein, respectively.
```

```
But there are different degree of overlapping within different proteins.*/
```

```
int npar_p_; /* number of elements in pmatrix, for 2-, 3-, and 4-sites,
there are 1, 3, 4 elements in pmatrix per protein, respectively.
```

```
But there are different degree of overlapping within different proteins.*/
```

```
int npar_R_; /* number of elements in R2matrix,
there are 2, 3, and 4 elements per residue per field for 2-, 3-, and 4-site model,
respectively.*/
```

```
int npar_w_; /* number of elements in dwmatrix,
there are 2, 3, and 4 elements per residue for 2-, 3-, and 4-site model, respectively.
*/
```

```
double kmatrix[MAXPAR]; /* kmatrix */
```

```
double pmatrix[MAXPAR]; /* pmatrix */
```

```
double R2matrix[MAXDATA]; /* R2matrix */
```

```
double dwmatrix[MAXDATA]; /* dwmatrix */
```

```

double kumatrix[MAXPAR]; /* uncertainty of kmatrix */
double pumatrix[MAXPAR]; /* uncertainty of pmatrix */
double R2umatrix[MAXDATA]; /* uncertainty of R2matrix */
double dwumatrix[MAXDATA]; /* uncertainty of dwmatrix */
double ddwmatrix[MAXDATA]; /* ddwmatrix */

int nres_; /* number of residue */
int nhsfrq_; /* number of field */
int nprot_; /* number of proteins */
}par;

/* Temperature dependent parameters common for all nuclei, residues and
frequencies */
struct temp_dep_values
{
int num_data_; /* number of data points at this temp */
double temp_; /* temperature */

double P_[3]; /* PA, PB and PC */
double eP_[3]; /* Errors of PA, PB and PC */
char fix_P_[3]; /* Fix flag for PA, PB and PC */

double KEX_[3]; /* KEX_AB, KEX_AC and KEX_BC */
double eKEX_[3]; /* Errors KEX_AB, KEX_AC and KEX_BC */
char fix_KEX_[3]; /* Fix flag for KEX_AB, KEX_AC and KEX_BC */

double cov_pk_[6][6]; /* covariance matrix for KEX and P variables */
}tdv[MAXPAR];

/* nucleus and residue dependent parameters common for all tempratures and
frequencies */
struct res_dep_values
{
int num_data_; /* number of data points at this nr and nucl */
int nr_; /* residue */
int nucl_; /* nucleus */
double xhgmrat_; /* ratio dfrq/hfrq */

int csord_[3]; /* order of polynomial to approximate CS(T) dependence */

double CS0_[3]; /* Chemical shift for states A, B, and C at T=0 */

```

```

double eCS0_[3]; /* Error in chemical shift for states A, B, and C at T=0 */
char fix_CS0_[3]; /* Fix flag for chemical shift for states A, B, and C at T=0 */

double CS1_[3]; /* First derivative of CS(T) at T=0 */
double eCS1_[3]; /* Error in first derivative of CS(T) at T=0 */
char fix_CS1_[3]; /* Fix flag for first derivative of CS(T) at T=0 */

double CS2_[3]; /* Second derivative of CS(T) at T=0 */
double eCS2_[3]; /* Error in second derivative of CS(T) at T=0 */
char fix_CS2_[3]; /* Fix flag for second derivative of CS(T) at T=0 */
}rdv[MAXPAR];

/* Residue, nucleus frequency, and temperature and coherence dependent
parameters */
struct freq_dep_values
{
int num_data_; /* number of data corresponding to such combination of
res,nucl,nucl1,coh,temp,hsfrq */
int nr_; /* residue */
int nucl_; /* main nucleus */
int nucl1_; /* additional nucleus */
double xhgmrat_; /* ratio dfrq/hfrq for main nucleus */
double xhgmrat1_; /* ratio dfrq/hfrq for additional nucleus */
int coh_; /* coherence */
double temp_; /* temperature */
double hsfrq_; /* 1H frequency */

int irdv_; /* index in structure rdv for main nucleus */
int irdv1_; /* index in structure rev for additional nucleus */

double W_[3]; /* Larmor frequencies for states A, B, and C for main nucleus */
double eW_[3]; /* Errors in Larmor frequencies for states A, B, and C for main
nucleus */
char fix_W_[3]; /* Fix flag for Larmor frequencies for states A, B, and C for
main nucleus */

double W1_[3]; /* Larmor frequencies for states A, B, and C for additional
nucleus */
double eW1_[3]; /* Errors in Larmor frequencies for states A, B, and C for
additional nucleus */

```

```
char fix_W1_[3]; /* Fix flag for Larmor frequencies for states A, B, and C for
additional nucleus */
```

```
double R1_[3]; /* R1 in states A, B, and C */
double eR1_[3]; /* Errors in R1 in states A, B, and C */
char fix_R1_[3]; /* Fix flag for R1 in states A, B, and C */
char glob_R1_; /* Glob flag for R1 in states A, B, and C */
```

```
double R2_[3]; /* R2 in states A, B, and C */
double eR2_[3]; /* Errors in R2 in states A, B, and C */
char fix_R2_[3]; /* Fix flag for R2 in states A, B, and C */
char glob_R2_; /* Glob flag for R2 in states A, B, and C */
}fdv[MAXDATA];
```

```
EXT struct mint
```

```
{
int nfcn_; /* number of functions (used data) */
int npar_; /* number of adjusted parameters */
char part_[MAXPAR]; /* types of adjusted parameters */
int res_[MAXPAR]; /* residues */
int coh_[MAXPAR]; /* coherence */
int nucl_[MAXPAR]; /* nucleus 1 types */
int nucl1_[MAXPAR]; /* nucleus 2 types */
double temp_[MAXPAR]; /* temperatures */
double hsrq_[MAXPAR]; /* fields */
int state_[MAXPAR]; /* state index for given parameter */
double parv_[MAXPAR]; /* value of adjusted parameter */
double exactInput_[MAXPAR]; /* Exact value of adjustable parameters */
double *parvp_[MAXPAR]; /* pointer to the value of adjusted parameter in
strucaures par tdv rdv fdv */
double *paru_[MAXPAR]; /* uncertainty of adjusted parameter */
int *df_; /* number of degrees of freedom */
double *F_; /* target function */
double *P_lvl_;

int npark_; /* Number of kinetics related adjustable parameters */
int nparp_; /* Number of population related adjustable parameters */
int nparR_; /* Number of R20 related adjustable parameters */
int nparw_; /* Number of dw related adjustable parameters */
```

```
double *parvpk_[MAXPAR]; /* pointers to the values of adjustable ks */
```

```

double *parvpp_[MAXPAR]; /* pointers to the values of adjustable ps */
double *parvpR_[MAXPAR]; /* pointers to the values of adjustable R2s */
double *parvpw_[MAXPAR]; /* pointers to the values of adjustable dws */

double *parvpku_[MAXPAR]; /* pointers to the uncertainties of adjustable ks */
double *parvppu_[MAXPAR]; /* pointers to the uncertainties of adjustable ps */
double *parvpRu_[MAXPAR]; /* pointers to the uncertainties of adjustable R2s
*/
double *parvpwu_[MAXPAR]; /* pointers to the uncertainties of adjustable dws
*/

int protein_[MAXPAR]; /* protein */
}mi;

FILE *input;

/* Functions */

int parser();
int set_def();
int init_kwords();
char* cutoff_comm(char *s);
int message(char *mess);
int kword(char *s);
char* find_kword(int s);
int help(char *s);
int word_comp(struct dict *s1,struct dict *s2);
int read_data(char *s);
int write_data(char *s);
int del_data(char *s);
int fix_par(char *s,char c);
int set_val(char *s);
int minimize(char *s);
int backup(char *s);
int restore(char *s);
char *rmalloc(size_t n);
void sort_data();
void sort_par();
void sort_tfr();
void del_dupl_data();
int add_par();

```

```

int del_unused_par();
void update_linked_par();
int clc_target(double *F);
int start_dnls1();
static double chi2(double *x);
double clc_P_chi2(double f,int m);
void fcn(int *iflag,int *M,int *N,double *x,double *fvec,double *fjac,int *ldfjac);
static int clc_fcn(double *fvec,double *x);
static int clc_jac(double *fjac,double *x);

int R24s(double *R2,double *W,double *P,double *KEX, double staField, double
tau, int nrep,double *R);
int R23s(double *R2,double *W,double *P,double *KEX, double staField, double
tau, int nrep,double *R);
int R22s(double *R2,double *W,double *P,double *KEX, double staField, double
tau, int nrep,double *R);
int R24smajor(double *R2,double *W,double *P,double *KEX, double staField,
double tau, int nrep,double *R);
int R23smajor(double *R2,double *W,double *P,double *KEX, double staField,
double tau, int nrep,double *R);
int R22smajor(double *R2,double *W,double *P,double *KEX, double staField,
double tau, int nrep,double *R);
int RexCR(double *R2,double *W,double *P,double *KEX, double staField,
double tau, int nrep,double *R);
int Rex3sM(double *R2,double *W,double *W1,double *P,double *KEX,double
time_T2,int nrep,double *R,int sim);
int Rex3R1R(double *R2,double *R1,double *W,double *P,double *KEX,double
B1,double OFF,double time_T2,double *R);
/*int RexCR(double R2A, double R2B, double DW, double DWH, double
PA,double KEX,double time_T2,int nrep,double *R,int sim);*/
int RexPF(double R1A,double R1B,double R2A,double R2B,double DW,double
PA,double KEX,double B1,double OFF,double time_T2,double *R);
#include "head_cp.h"
#include "slatec.h"
#include <stdlib.h>

int read_data(s)
char *s;
{
FILE *ifp;
unsigned int seed;

```

```

int i, j, k, p, q, itmp, file_f, u1, u2;
double u3, u4, x1;
char s1[MAXSTR], s2[MAXSTR], s3[MAXSTR], s4[MAXSTR], s5[MAXSTR],
s6[MAXSTR], s7[MAXSTR], s8[MAXSTR], s9[MAXSTR], s10[MAXSTR],
s11[MAXSTR], s12[MAXSTR], s13[MAXSTR], s14[MAXSTR],
s15[MAXSTR], s16[MAXSTR], s17[MAXSTR], s18[MAXSTR],
s19[MAXSTR], s20[MAXSTR];
char file[MAXSTR], type[MAXSTR], line[MAXSTR];

if(2>(itmp=sscanf(s, "%*s%s%s", file, type)))
{
fprintf(stderr, "use %s <file name> <type (d-data) or (k-kex) or (p-populations) or
(R-R2s) or (w-deltaW) or (n-noise)>\n", find_keyword(READ));
return(1);
}

if(type[0]!='d' && type[0]!='k' && type[0]!='p' && type[0]!='R' && type[0]!='w'
&& type[0]!='n')
{
fprintf(stderr, "use %s <file name> <type (d-data) or (k-kex) or (p-populations) or
(R-R2s) or (w-deltaW) or (n-noise)>\n", find_keyword(READ));
return(1);
}

if(NULL==(ifp=fopen(file,"r"))){fprintf(stderr, "can not open file %s\n",
file);return(1);}

file_f=1; /*do not know what is this and why*/

/* start reading data. Data has to be read first because there is necessary
information for parameter reading*/
if(type[0]=='d')
{
j=0;
k=0;
p=0;
q=0;
while(NULL!=fgets(line, MAXSTR, ifp))
{

```

```

        i=par.ndata_; /* By now, the parameters were actually filled as
default, 0*/
        if(1!=sscanf(line, "%d%d%d%d%d%d%lf%lf%lf%lf",
&dat[i].mol_, &dat[i].contimIn_, &dat[i].temperatureIn_, &dat[i].staFieldIn_,
&dat[i].proteinIn_, &dat[i].residueIn_, &dat[i].ncyc_, &dat[i].staField_,
&dat[i].tau_, &dat[i].val_, &dat[i].err_))
        {
                fprintf(stderr, "The data file at line %d is not right\n", i);
/*Normally this will not happen because I will write a perl script to check this
kind of error in the data file*/
                return(-1);
        }

        if(dat[i].proteinIn_ > p){p=dat[i].proteinIn_;} /* Make sure the
proteinIn is increasing */
        if(dat[i].residueIn_ > j){j=dat[i].residueIn_;} /* Make sure the
residueIn is increasing */
        if(dat[i].staFieldIn_ > q){q=dat[i].staFieldIn_;k++;} /* Make sure
the fieldIn is increasing */

        par.ndata_++; /* count the number of data point regardless residue,
field, protein, etc. */
        }/*Last braket for while*/

        par.nprot_=p; /* Count how many proteins used given that protein Id is
consecutive from 1.*/
        par.nres_=j; /* Count how many residues involved */
        par.nhsfrq_=k; /* Count how many static fields are used */
        printf("There are such number of proteins, number of residues, number of
static field, and number of total data points: %d %d %d %d\n", par.nprot_,
par.nres_, par.nhsfrq_, par.ndata_);
} /*Last braket for if(type[0]='d') */

if(type[0]=='n')
{
        if(par.ndata_<1){printf("There is not data loaded\n");return(1);}
        for(i=0;i<par.ndata_;i++) {
                u1=rand();
                u2=rand();
                u3=u1*1.0/(1.0*RAND_MAX+1.0);
                u4=u2*1.0/(1.0*RAND_MAX+1.0);

```

```

        x1=sqrt(-2*log(u3))*cos(2*pi*u4);
        dat[i].err_=x1*dat[i].val_/1000.0;
    }
}

/* start reading kinetics parameters assuming the data has been read already */

if(type[0]=='k')
{

if(par.ndata_==0){fprintf(stderr, "Read data first.\n");return(1);}

mi.npark_=0; /* number of kinetics related adjustable parameters */
i=0;
j=0;

/* For two-state model, there is one KEX per protein. Each residue from the same
protein shares the same kex.*/
/* I am not worrying about two-state equilibrium because we have evidence that
there are more than two states */
if (dat[0].mol_== 2 || dat[0].mol_== 31 || dat[0].mol_== 32)
{
    while(NULL!=fgets(line, MAXSTR, ifp))
    {
        if(2!=sscanf(line, "%lf%lf", &par.kmatrix[j], &par.kumatrix[j]))
        {fprintf(stderr, "one kex and one uncertainty per line and no flags
showing fixed or adjustable!\n");return(1);} /*Obviously, it has to be adjustable.
No need for flag indicating adjustable or not.*/

        printf("%lf %lf\n", par.kmatrix[j], par.kumatrix[j]);

        i++; /* Count how many proteins used */
        par.kmatrix[j] = par.kmatrix[j]*2.302585093;
        par.kumatrix[j] = par.kumatrix[j]*2.302585093;
        mi.parvpk_[mi.npark_]=&par.kmatrix[j];
        mi.parvpku_[mi.npark_]=&par.kumatrix[j];
        mi.npark_++;
        j++;
    }
    par.nke_=j;
}
}

```



```

        mi.npark_++;
    }

    if(s2[0]=='u'){
        mi.parvpk_[mi.npark_]=&par.kmatrix[j+1];
        mi.parvpku_[mi.npark_]=&par.kumatrix[j+1];
        mi.npark_++;
    } /* Most likely it will be assumed known from DH only construct
*/

    if(s3[0]=='u'){
        mi.parvpk_[mi.npark_]=&par.kmatrix[j+2];
        mi.parvpku_[mi.npark_]=&par.kumatrix[j+2];
        mi.npark_++;
    } /* This will be fixed to zero for linear four-state case */

    if(s4[0]=='u'){
        mi.parvpk_[mi.npark_]=&par.kmatrix[j+3];
        mi.parvpku_[mi.npark_]=&par.kumatrix[j+3];
        mi.npark_++;
    }

    j=j+4;

}
par.nke_=j;
}
if(i!=par.nprot_){fprintf(stderr, "Number of line in kmatrix.txt does not match the
number of proteins counted.\n");return(1);}
}

/* start reading population parameters assuming the data has been read already */

if(type[0]=='p')
{

if(par.ndata_==0){fprintf(stderr, "Read data first.\n");return(1);}
mi.nparp_=0; /* Index for population related adjustable parameters */
i=0;
j=0;

```



```

        printf("%lf %lf %lf %lf %lf %lf %lf %lf\n", par.pmatrix[j+0],
par.pumatrix[j+0], par.pmatrix[j+1], par.pumatrix[j+1], par.pmatrix[j+2],
par.pumatrix[j+2], par.pmatrix[j+3], par.pumatrix[j+3]);
        i++; /* Count how many number of proteins used */
        if(s1[0]=='u'){
            mi.parvpp_[mi.nparp_]=&par.pmatrix[j+0];
            mi.parvppu_[mi.nparp_]=&par.pumatrix[j+0];
            mi.nparp_++;
        }

        if(s2[0]=='u'){
            mi.parvpp_[mi.nparp_]=&par.pmatrix[j+1];
            mi.parvppu_[mi.nparp_]=&par.pumatrix[j+1];
            mi.nparp_++;
        } /* This will be fixed */

        if(s3[0]=='u'){
            mi.parvpp_[mi.nparp_]=&par.pmatrix[j+2];
            mi.parvppu_[mi.nparp_]=&par.pumatrix[j+2];
            mi.nparp_++;
        } /* This will also be fixed for the same reasons */

        if(s4[0]=='u'){
            mi.parvpp_[mi.nparp_]=&par.pmatrix[j+3];
            mi.parvppu_[mi.nparp_]=&par.pumatrix[j+3];
            mi.nparp_++;
        } /* This will be fixed and depend on the value of par.pmatrix[j] */

        j=j+4;}}
    par.npo_=j;
    if(i!=par.nprot_){fprintf(stderr, "Number of line in pmatrx.txt does not match the
number of proteins counted.\n");}}

/* start reading R2null parameters assuming the data has been read already. There
two R2null per residue per static field only for every model */

if(type[0]=='R'){
if(par.ndata_==0){fprintf(stderr, "Read data first.\n");return(1);}
mi.nparR_=0;
i=0;
j=0;

```



```

    mi.parvpRu_[mi.nparR_]=&par.R2umatrix[j+4];
    mi.nparR_++;
}
if(s6[0]=='u'){
    mi.parvpR_[mi.nparR_]=&par.R2matrix[j+5];
    mi.parvpRu_[mi.nparR_]=&par.R2umatrix[j+5];
    mi.nparR_++;
}
j=j+6;
}
par.nr2_=j;
if(i!=par.nres_*par.nhsfrq_) {fprintf(stderr, "Number of line in R2matrix.txt does
not match the number of residues time number of static fields.\n");}
}

/* start reading chemical shift parameters in ppm assuming the data has been read
already */

if(type[0]=='w')
{
if(par.ndata_==0){fprintf(stderr, "Read data first.\n");return(1);}

mi.nparw_=0;
i=0;
j=0;

/* For two-state model, there is only one dw. Another one will be set to zero in
the module of R2 calculation. */

if (dat[0].mol_== 2 || dat[0].mol_== 31 || dat[0].mol_== 32)
{
    while(NULL!=fgets(line, MAXSTR, ifp))
    {
        if(2!=sscanf(line, "%lf%lf", &par.dwmatrix[j], &par.dwumatrix[j]))
        {
            printf("%lf %lf", par.dwmatrix[j], par.dwumatrix[j]);
            fprintf(stderr, "one dw and one uncertainty of dw per line and no
flags showing fixed or adjustable!\n");return(1);
        }
        printf("%lf %lf\n", par.dwmatrix[j], par.dwumatrix[j]);
        i++; /* Count how many residues used */
    }
}
}

```



```

        mi.parpvwu_[mi.nparw_]=&par.dwumatrix[j+3];
        mi.nparw_++;
    }

    j=j+4;
}
par.nw_=j;
}

if(i!=par.nres_){fprintf(stderr, "Number of line in dwmatrix.txt does not match
the number of residues used.\n");}
}

}/*this is the last curly branket*/
#include <stdlib.h>
#include "head_cp.h"
#include "slatec.h"

#define DEBUG_
#define COVCS /* uncomment this to print covariance matrix for delta-cs to file
covcs.res */

int minimize(s)
char *s;
{
int i, nstarts;
if(1!=sscanf(s,"%*s%d",&nstarts)){nstarts=-1;} /* We have a choice to input
maximum number of function calls */

FILE *ifp, *ifpd, *ifpk, *ifpp, *ifpr, *ifpw;
char file[MAXSTR], s1[MAXSTR];
int itmp, file_f, j, k, k1, k2, k3, k4, p1, p2, p3, w1, w2;
double uncertainty[MAXPAR];

if(1!=(itmp=sscanf(s, "%*s%s%s", file, s1))){fprintf(stderr, "use: \n%s <file name>
p\n", find_kword(MIN));return(1);}

if(NULL==(ifp=fopen(file, "w"))){fprintf(stderr, "cannot open file
%s\n", file);return(1);}
if(NULL==(ifpd=fopen("calculateddata", "w"))){fprintf(stderr, "cannot open file
calculateddata\n");return(1);}

```

```

if(NULL==(ifpk=fopen("calculatedk","w"))){fprintf(stderr,"cannot open file
calculatedk\n");return(1);}
if(NULL==(ifpp=fopen("calculatedp","w"))){fprintf(stderr,"cannot open file
calculatedp\n");return(1);}
if(NULL==(ifpr=fopen("calculatedr","w"))){fprintf(stderr,"cannot open file
calculatedr\n");return(1);}
if(NULL==(ifpw=fopen("calculatedw","w"))){fprintf(stderr,"cannot open file
calculatedw\n");return(1);}

fprintf(stdout, "mi.npark_ is %d mi.nparR_ is %d mi.nparp_ is %d mi.nparw_ is
%d\n",mi.npark_, mi.nparR_, mi.nparp_, mi.nparw_);
fprintf(stdout, "par.nke_ is %d par.npo_ is %d par.nw_ is %d par.nr2_ is
%d\n",par.nke_, par.npo_, par.nw_, par.nr2_);
/*fprintf(ifp, "mi.npark_ is %d mi.nparR_ is %d mi.nparp_ is %d mi.nparw_ is
%d\n",mi.npark_, mi.nparR_, mi.nparp_, mi.nparw_);*/

if(par.ndata_==0 && par.npark_==0 && par.nparp_==0 && par.nparR_==0 &&
par.nparw_==0)
{fprintf(stderr, "Make sure read data.txt, kmatrix.txt, pmatrix.txt, R2matrix.txt
and dwmatrix.txt.\n");return(1);}

mi.nfcn_ = par.ndata_;           /* Generated when reading data*/
mi.npar_ = 0;                    /* Number of adjustable parameters, Start
with 0 */
mi.df_ = &(par.df_);            /* Still 0 */
mi.P_lvl_ = &(par.P_lvl_);      /* Still 0 */
mi.F_ = &(par.F_);              /* Still 0 */

for(i=0;i<mi.npark_;i++){
    mi.part_[mi.npar_]='k';      /* the kind of adjustable
parameters */
    mi.parvp_[mi.npar_]=mi.parvpk_[i]; /* Fill in pointer vector
pointing to adjustable parameters */
    mi.paru_[mi.npar_]=mi.parvpku_[i]; /* Fill in pointer vector pointing to
uncertainty of adjustable parameters */
    mi.npar_++;
}
for(i=0;i<mi.nparp_;i++){
    mi.part_[mi.npar_]='p';
    mi.parvp_[mi.npar_]=mi.parvpp_[i]; /* Fill in pointer vector pointing to
adjustable parameters */

```

```

    mi.paru_[mi.npar_]=mi.parvppu_[i]; /* Fill in pointer vector pointing to
uncertainty of adjustable parameters */
    mi.npar_++;
}
for(i=0;i<mi.nparw_;i++){
    mi.part_[mi.npar_]='w';
    mi.parvp_[mi.npar_]=mi.parvpw_[i]; /* Fill in pointer vector pointing to
adjustable parameters */
    mi.paru_[mi.npar_]=mi.parvpwu_[i]; /* Fill in pointer vector pointing to
uncertainty of adjustable parameters */
    mi.npar_++;
}
for(i=0;i<mi.nparR_;i++){
    mi.part_[mi.npar_]='R';
    mi.parvp_[mi.npar_]=mi.parvpR_[i]; /* Fill in pointer vector pointing to
adjustable parameters */
    mi.paru_[mi.npar_]=mi.parvpRu_[i]; /* Fill in pointer vector pointing to
uncertainty of adjustable parameters */
    mi.npar_++;
}

/* Therefore in the adjustable parameter vector, kinetics, thermodynamics, R2
null and chemical shift info are in stored in such an order*/
if(mi.npar_==0){fprintf(stderr, "Nothing to minimize.\n");return(-1);}
for(i=0;i<mi.npar_;i++)
{mi.parv_[i]=*(mi.parvp_[i]);
mi.exactInput_[i]=*(mi.parvp_[i]);
fprintf(stdout, "Adjustables: %lf %lf\n", mi.parv_[i], mi.exactInput_[i]);
fprintf(ifp, "%lf %lf\n", mi.parv_[i], mi.exactInput_[i]);}

/* see SLATEC documentaion for function dnls1 for more details */
double uuu, uu, F, t, tmin; /* F is the value of the targeted chiSquare function */
int a, u, l, m, iopt, M, N, LDFJAC, MODE, NPRINT, INF, NFEV, NJEV, *IPVT,
MAXFEV;
double *FVEC; /* functions evaluated at the output X */
double *FJAC, *DIAG, FTOL, XTOL, GTOL, *QTF, *WA1, *WA2, *WA3,
*WA4, EPSFCN, FACTOR, *R;
iopt=1; /* 2 or 3: we have to provide the jacobian */
M=mi.nfcn_; /* number of functions, i.e. number of data points which is counted
when reading data*/
N=mi.npar_; /* number of adjustable variables which is determined just now*/

```

```

LDFJAC=M;
FTOL=1e-8; XTOL=1.0e-15; GTOL=0.0; /* tolerance for optimizations */

if(nstarts<1){MAXFEV=20000*mi.npar_; fprintf(stdout,"Max. number of
function evaluation (default): %d\n",MAXFEV);}
else{MAXFEV=nstarts; fprintf(stdout,"Max. number of function evaluation:
%d\n",MAXFEV);}

NPRINT=1;
EPSFCN=EPSFCN_DEF; MODE=1; FACTOR=100.;

FVEC=FJAC=R=DIAG=QTF=WA1=WA2=WA3=WA4=NULL;
IPVT=NULL;

/* Determine whether there are enough memory, Work it out later */

if(NULL==(FVEC=(double*)rmalloc(sizeof(double)*M))
|| NULL==(FJAC=(double*)rmalloc(sizeof(double)*M*N))
|| NULL==(R=(double*)rmalloc(sizeof(double)*M*N))
|| NULL==(DIAG=(double*)rmalloc(sizeof(double)*N))
|| NULL==(QTF=(double*)rmalloc(sizeof(double)*N))
|| NULL==(IPVT=(int*)rmalloc(sizeof(int)*N))
|| NULL==(WA1=(double*)rmalloc(sizeof(double)*N))
|| NULL==(WA2=(double*)rmalloc(sizeof(double)*N))
|| NULL==(WA3=(double*)rmalloc(sizeof(double)*N))
|| NULL==(WA4=(double*)rmalloc(sizeof(double)*M))) {
fprintf(stderr,"Not enough memory\n"); return(-1);}

/* Record the estimated guess of each adjustable parameters in the output file*/
for(i=0;i<mi.npar_-mi.nparR_;i++){mi.parv_[i]=*(mi.parvp_[i]);}
for(i=mi.npar_-mi.nparR_;i<mi.npar_;i++){mi.parv_[i]=mi.exactInput_[i];}

for(i=0;i<mi.npar_;i++)
{fprintf(stdout, "%lf %lf\n",mi.parv_[i], *(mi.paru_[i]));}
fprintf(stdout, "Start dnls1_\n");
dnls1_(fcn, &iopt, &M, &N, (&mi)->parv_, FVEC, FJAC, &LDFJAC, &FTOL,
&XTOL, &GTOL, &MAXFEV, &EPSFCN, DIAG, &MODE, &FACTOR,
&NPRINT, &INF, &NFEV, &NJEV, IPVT, QTF, WA1, WA2, WA3, WA4);

/* Record resulting value of each adjustable parameter in the output file*/
for(i=0;i<mi.npar_;i++){

```



```

    fprintf(stdout, "conditions for INF=1 and INF=3 both hold.\n");break;
case 4:
    fprintf(stdout, "the cosine of the angle between FVEC and any column of the
jacobian is at most GTOL in absolute value.\n");break;
case 5:
    fprintf(stdout, "number of calls to FCN for function evaluation has
reached.\n");break;
default:
    if(abrflg){fprintf(stdout, "Minimization aborted\n");abrflg=0;break;}
    fprintf(stdout, "Error: Minimization failed;INF=%d",INF);
    par.clc_ =FALSE;
    par.clcu_ =FALSE;
    return(1);
}

for(i=0;i<mi.npar_;i++) (*(mi.parvp_[i]))=mi.parv_[i];
*(mi.F_)=0;
*(mi.df_)=0;
*(mi.df_)=(mi.nfcn_-mi.npar_-mi.npar_);

for(i=0;i<mi.nfcn_;i++)
{*(mi.F_)+=FVEC[i]*FVEC[i];if(i>mi.nfcn_-mi.npar_ && FVEC[i]>0)
*(mi.df_)+++;}

if(*(mi.F_)>50*(*(mi.df_))){*(mi.P_lvl_)=0;}
else{*(mi.P_lvl_)=clc_P_chi2(*(mi.F_),*(mi.df_));}

fprintf(stdout,"Total F=%.4g\nCHI2 probability %.4g for %d degrees of
freedom\n",*(mi.F_),*(mi.P_lvl_),*(mi.df_));

/* calculate covariance matrix to find uncertanties of parameters */
dcov_(fcn,&iopt,&(mi.nfcn_),&(mi.npar_),(&mi)-
>parv_,FVEC,R,&(mi.nfcn_),&INF,WA1,WA2,WA3,WA4);

par.clcu_ =TRUE;
switch(INF){
case 0:
    fprintf(stderr,"Improper input parameters\n");par.clcu_ =FALSE; return(1);
    break;
case 1:
    fprintf(stdout,"Uncertainties were successfully calculated\n");

```

```

for(i=0;i<mi.npar_;i++)(*mi.paru_[i])=sqrt(*(R+mi.nfcn_*i+i));

/*#ifdef COVCS
if(NULL==(ifp=fopen("covcs.res","w"))){
fprintf(stderr,"can't open file %s\n","covcs.res");break;}
for(i=0;i<mi.npar_;i++)
for(j=0;j<i;j++){
if(mi.part_[i]=='c' && mi.part_[j]=='c' && mi.res_[i]==mi.res_[j]){
fprintf(ifp,"%5d nucl: %d state: %d %10.6lf %10.6lf   nucl: %d state: %d
%10.6lf %10.6lf cov:
%10.6lf\n",mi.res_[i],mi.nucl_[i],mi.state_[i],*(mi.parvp_[i]),*(mi.paru_[i]),mi.n
ucl_[j],mi.state_[j],*(mi.parvp_[j]),*(mi.paru_[j]),*(R+mi.nfcn_*i+j))/(*(mi.paru
_[i]))/(*(mi.paru_[j])) );
}}
fclose(ifp);
#endif*/

break;
case 2:
fprintf(stdout,"Uncertainties calculation failed\n");par.clcu_=FALSE; return(1);
break;}

for(i=0;i<mi.npar_;i++)
{
fprintf(ifp, "%lf %lf\n", mi.parv_[i], *(mi.paru_[i]));
fprintf(stdout, "%lf %lf\n", mi.parv_[i], *(mi.paru_[i]));
}
fprintf(stdout, "\n");

free(FVEC); free(FJAC); free(DIAG); free(QTF); free(IPVT); free(R);
free(WA1); free(WA2); free(WA3); free(WA4);

fclose(ifp);
fprintf(stdout, "\n");
return(0);
} /* The end bracket for function start_dnls1. */

void fcn(int *iflag, int *M, int *N, double *X, double *fvec, double *fjac, int
*ldfjac)
{
int i;

```

```

double t;

if(abrflg){*iflag=-1; return;}
switch(*iflag){
  case 0:
    t=0;
    for(i=0;i<mi.nfcn_;i++) {t+=fvec[i]*fvec[i];}
    fprintf(stdout, "F = %12.8g\n", t);
    #ifdef DEBUG
      for(i=0;i<mi.npar_;i++)fprintf(stdout, "%10c",mi.part_[i]);
      fprintf(stdout, "\n");
      for(i=0;i<mi.npar_;i++)fprintf(stdout, "%10.6g",mi.parv_[i]);
      fprintf(stdout, "\n");
      for(i=0;i<mi.npar_;i++)fprintf(stdout, "%10.6g",fvec[i]);
      fprintf(stdout, "\n\n");
    #endif
    break;
  case 1:
    /* t=0;
    for(i=0;i<mi.nfcn_;i++) {t+=fvec[i]*fvec[i];}
    fprintf(stdout, "F = %12.8g\n", t);*/
    /*fprintf(stdout, "TEST\n");*/
    clc_fcn(fvec, X);
    break;
  case 2:
    clc_jac(fjac, X);
    break;
  case 3:
    break;
  default:
    fprintf(stderr, "Fatal error\n");
    exit(1);
}
} /* The end branket for function fcn. */

static int clc_jac(double *fjac, double *x)
{printf("Test in clc_jac. It is not required in current cases\n");}

static int clc_fcn(double *fvec, double *x)
{
register int i;

```

```

for(i=0;i<mi.npar_;i++) (*(mi.parvp_[i]))=x[i];
if(0!=clc_target(&(par.F_))){fprintf(stderr, "error during model values
calculation\n");}
for(i=0;i<mi.nfcn_;i++) fvec[i]=(dat[i].val_-dat[i].cval_)/dat[i].err_;
for(i=0;i<mi.npar_;i++){
  if(mi.part_[i]=='p' && x[i]<=0){*(fvec+i+mi.nfcn_-
mi.npar_)+=x[i]*x[i]*P_COEFF;}
  if(mi.part_[i]=='p' && x[i]>=1.0){*(fvec+i+mi.nfcn_-mi.npar_)+=(1.0-
x[i])*(1.0-x[i])*P_COEFF;}
  if(mi.part_[i]=='k' && x[i]<=0){*(fvec+i+mi.nfcn_-
mi.npar_)+=x[i]*x[i]*K_COEFF;}
  if(mi.part_[i]=='R' && x[i]<=0){*(fvec+i+mi.nfcn_-
mi.npar_)+=x[i]*x[i]*R_COEFF;}
  if(mi.part_[i]=='R' && x[i]>50){*(fvec+i+mi.nfcn_-mi.npar_)+=(x[i]-50)*(x[i]-
50)*R_COEFF;}
}
return(0);
} /* The end branket for function clc_fcn. */

int clc_target(double *F)
{
register int i, j, a, b, c, d, dd;
double R, R2matrix, dwmatrix, pmatrix, kmatrix;
*F=0;
for(i=0;i<mi.nfcn_;i++)
{
  if (dat[0].mol_==7111) {
    if (dat[i].mol_!=71){fprintf(stderr, "wrong model index in at least some data
points\n");return(1);}
    a=3*4*dat[i].proteinIn_-3*4+4*dat[i].temperatureIn_-4;
    par.kmatrix[a+1]=par.kmatrix[4*dat[i].temperatureIn_-4+1];
    par.kmatrix[a+3]=par.kmatrix[4*dat[i].temperatureIn_-4+3];
    if (dat[i].proteinIn_==1){
      par.pmatrix[a+1]=0.95*(1.0-par.pmatrix[a+0]-par.pmatrix[a+3]);
      par.pmatrix[a+2]=0.05*(1.0-par.pmatrix[a+0]-par.pmatrix[a+3]);
    }
    else {
par.pmatrix[a+3]=par.pmatrix[a+0]*par.pmatrix[3+4*dat[i].temperatureIn_-
4]/par.pmatrix[0+4*dat[i].temperatureIn_-4];
par.pmatrix[a+1]=(1.0-par.pmatrix[a+3]-
par.pmatrix[a])*par.pmatrix[1+4*dat[i].temperatureIn_-

```

```

4]/(par.pmatrix[2+4*dat[i].temperatureIn_ -
4]+par.pmatrix[1+4*dat[i].temperatureIn_-4]);
par.pmatrix[a+2]=(1.0-par.pmatrix[a+3]-
par.pmatrix[a])*par.pmatrix[2+4*dat[i].temperatureIn_ -
4]/(par.pmatrix[2+4*dat[i].temperatureIn_ -
4]+par.pmatrix[1+4*dat[i].temperatureIn_-4]);
}
b=4*dat[i].residueIn_-4;
if(par.dwmatrix[b]==0.0 && par.dwmatrix[b+3]==0.0)
par.dwmatrix[b+2]=par.dwmatrix[b+1];
if(par.dwmatrix[b+3]==0.0 && par.dwmatrix[b+2]==0.0)
par.dwmatrix[b+1]=par.dwmatrix[b+0];
/* c=4*par.nhsfrq_*dat[i].residueIn_+4*dat[i].staFieldIn_+2*dat[i].coh_-
4*par.nhsfrq_-4-2;*/
c=186*dat[i].proteinIn_-186+36*dat[i].residueIn_-36 +
6*2*dat[i].temperatureIn_-12 + 6*dat[i].contimIn_-6+2*dat[i].staFieldIn_-2;

if(0!=R24s(&par.R2matrix[c],&par.dwmatrix[b],&par.pmatrix[a],&par.kmatrix[a
],dat[i].staField_, dat[i].tau_,dat[i].ncyc_,&R))
{fprintf(stderr,"error in R24S\n");return(1);}
dat[i].cval_=R;
dat[i].F_=(dat[i].val_-dat[i].cval_)*(dat[i].val_-
dat[i].cval_)/(dat[i].err_*dat[i].err_);
(*F)+=dat[i].F_;
}
else if (dat[0].mol_==71){
if (dat[i].mol_!=71){fprintf(stderr,"wrong model index in at least some data
points\n");return(1);}
a=4*(6*dat[i].proteinIn_-6+2*dat[i].temperatureIn_-2+dat[i].contimIn_-1);
if(dat[i].proteinIn_<5)
{par.kmatrix[a+1]=par.kmatrix[4*24+4*(2*dat[i].temperatureIn_ -
2)+4*(dat[i].contimIn_-1)+1];}
if(dat[i].proteinIn_==2 || dat[i].proteinIn_==3 ||
dat[i].proteinIn_==4){par.kmatrix[a+3]=par.kmatrix[4*(2*dat[i].temperatureIn_ -
2)+4*(dat[i].contimIn_-1)+3];}
if (dat[i].proteinIn_==1){
d=4*6*4+4*2*dat[i].temperatureIn_-8+4*dat[i].contimIn_-4;
par.pmatrix[a+1]=(1.0-par.pmatrix[a+3]-par.pmatrix[a])*par.pmatrix[d+1];
par.pmatrix[a+2]=(1.0-par.pmatrix[a+3]-par.pmatrix[a])*(1.0-
par.pmatrix[d+1]);
}
}

```

```

else {
    d=4*6*4+4*2*dat[i].temperatureIn_-8+4*dat[i].contimIn_-4;
    par.pmatrix[a+3]=par.pmatrix[a]*par.pmatrix[4*2*dat[i].temperatureIn_-
8+4*dat[i].contimIn_-4+3]/par.pmatrix[4*2*dat[i].temperatureIn_-
8+4*dat[i].contimIn_-4+0];
    par.pmatrix[a+1]=(1.0-par.pmatrix[a+3]-par.pmatrix[a])*par.pmatrix[d+1];
    par.pmatrix[a+2]=(1.0-par.pmatrix[a+3]-par.pmatrix[a]*(1.0-
par.pmatrix[d+1]));
}
b=4*2*dat[i].residueIn_-8+4*dat[i].contimIn_-4;
dd=3*2*dat[i].residueIn_-6+3*dat[i].contimIn_-3;
if(par.ddwmatrix[dd+dat[i].temperatureIn_-1]!=0.0){

if(par.dwmatrix[b+0]<par.pmatrix[a+2]*par.dwmatrix[b+2]/(par.pmatrix[a+1]+pa
r.pmatrix[a+2])){par.dwmatrix[b+3]=((par.pmatrix[a+0]+par.pmatrix[a+3])/par.p
matrix[a+3])*par.pmatrix[a+2]*par.dwmatrix[b+2]/(par.pmatrix[a+1]+par.pmatr
ix[a+2])-par.ddwmatrix[dd+dat[i].temperatureIn_-1]-
par.dwmatrix[b+0]*par.pmatrix[a+0]/(par.pmatrix[a+0]+par.pmatrix[a+3]));}

else {par.dwmatrix[b+3]=((par.pmatrix[a+0]+par.pmatrix[a+3])/par.pmatrix[a+3])
*(par.pmatrix[a+2]*par.dwmatrix[b+2]/(par.pmatrix[a+1]+par.pmatrix[a+2])+par.
ddwmatrix[dd+dat[i].temperatureIn_-1]-
par.dwmatrix[b+0]*par.pmatrix[a+0]/(par.pmatrix[a+0]+par.pmatrix[a+3]));}
}
c=6*186*dat[i].proteinIn_-6*186+36*dat[i].residueIn_-36 +
12*dat[i].temperatureIn_-12 + 6*dat[i].contimIn_-6+2*dat[i].staFieldIn_-2;

if(0!=R24s(&par.R2matrix[c],&par.dwmatrix[b],&par.pmatrix[a],&par.kmatrix[a
],dat[i].staField_, dat[i].tau_,dat[i].ncyc_,&R))
    {fprintf(stderr,"error in R24S\n");return(1);}
    dat[i].cval_=R;
    dat[i].residual_=dat[i].val_-dat[i].cval_;
    dat[i].F_=(dat[i].residual_*dat[i].residual_)/(dat[i].err_*dat[i].err_);
    (*F)+=dat[i].F_;
}
else if (dat[0].mol_==51) {
    if (dat[i].mol_!=51){fprintf(stderr,"wrong model index in at least some data
points\n");return(1);}
    a=3*dat[i].proteinIn_-3; /* Indexing which protein is this data point belonging
to */
    par.kmatrix[a+1]=par.kmatrix[1];

```

```

    if (a==0){par.pmatrix[2]=1.0-par.pmatrix[1]-par.pmatrix[0];}
    else {
        par.pmatrix[a+2]=(1.0-
par.pmatrix[a])*par.pmatrix[2]/(par.pmatrix[2]+par.pmatrix[1]);
        par.pmatrix[a+1]=(1.0-
par.pmatrix[a])*par.pmatrix[1]/(par.pmatrix[2]+par.pmatrix[1]);
    }
    b=3*dat[i].residueIn_-3; /* Indexing which residue is this data point belonging
to */
    c=4*par.nhsfrq_*dat[i].residueIn_+4*dat[i].staFieldIn_+2*dat[i].coh_-
4*par.nhsfrq_-4-2; /* Indexing which R2 null to use because it varies with both
residues and fields */

if(0!=R23s(&par.R2matrix[c],&par.dwmatrix[b],&par.pmatrix[a],&par.kmatrix[a
],dat[i].staField_, dat[i].tau_,dat[i].ncyc_,&R))
    {fprintf(stderr,"error in R23S\n");return(1);}
    dat[i].cval_=R;
    dat[i].F_=(dat[i].err_+dat[i].val_-dat[i].cval_)*(dat[i].err_+dat[i].val_-
dat[i].cval_)/(dat[i].err_*dat[i].err_);
    (*F)+=dat[i].F_;
}
else if (dat[0].mol_ == 2) {
    if (dat[i].mol_ != 2){fprintf(stderr,"wrong model index in at least some data
points\n");return(1);}
    a=dat[i].proteinIn_-1; /* Indexing which protein is this data point belonging to
*/
    par.pmatrix[a+1]=1.0 - par.pmatrix[a];
    b=dat[i].residueIn_-1; /* Indexing which residue is this data point belonging to
*/
    c=2*par.nhsfrq_*dat[i].residueIn_+2*dat[i].staFieldIn_+dat[i].coh_-
2*par.nhsfrq_-3; /* Indexing which R2 null to use because it varies with both
residues and fields */

if(0!=R22s(&par.R2matrix[c],&par.dwmatrix[b],&par.pmatrix[a],&par.kmatrix[a
],dat[i].staField_, dat[i].tau_,dat[i].ncyc_,&R))
    {fprintf(stderr,"error in R22S\n");return(1);}
    dat[i].cval_=R;
    dat[i].F_=(dat[i].err_+dat[i].val_-dat[i].cval_)*(dat[i].err_+dat[i].val_-
dat[i].cval_)/(dat[i].err_*dat[i].err_);
    (*F)+=dat[i].F_;
}
}

```

```

else
  {fprintf(stderr,"wrong model index, using 1_flat, 2_CR, 3_two sites, 4_linear
three sites, 5_circular three sites, 6_linear four sites, 7_circular four sites only");
return(1);}
}
return(0);
} /* The end branket for function clc_target */
#include <stdlib.h>
#include "head_cp.h"
#include "slatec.h"

static double M_freedom;

static double chi2(double *x)
{
int IERR;
double m,lgam;

m=0.5*M_freedom;
lgam=dgamln_(&m,&IERR);
if(IERR){
fprintf(stderr,"Warning: SLATEC function DGAMLN returned
IERR=%d\n",IERR);
return(0);}

return(exp( (m-1.)*log(*x)-(*x)*0.5-m*log(2.)-lgam ));
}

double clc_P_chi2(double f,int m)
/* chi2 probability to get loss F > f by chance */
/* f - value of chi2 loss function */
/* m - # degrees of freedom */
{
double RESULT, ABSERR, a, b, EPSABS, EPSREL;
double WORK[400];
int NEVAL,IER,IWORK[100],LIMIT,LENW,LAST;

if(f>m*50) return(0.0);
a=1.e-20; b=f+2.*a; M_freedom=m;
EPSABS=EPSREL=1.e-8;
LIMIT=100; LENW=400;

```

```

dqags_(chi2,&a,&b,&EPSABS,&EPSREL,&RESULT,
&ABSERR,&NEVAL,&IER,&LIMIT,&LENW, &LAST, IWORK, WORK);
if(IER){fprintf(stderr,"Warning: SLATEC function DQAGS returned
IER=%d\n",IER);}
if(1.-RESULT > 0.0)return(1.-RESULT);
else return(0);
}

#define MAIN
#include "head_cp.h"
#include "slatec.h"

main()
{
/*Make sure to run this program in the right directory*/
/*if(NULL==getenv("CPMG_FIT_DIR")) strcpy(cpmg_fit_dir, "");
else strcpy(cpmg_fit_dir, getenv("CPMG_FIT_DIR"));
if(strlen(cpmg_fit_dir)
if(cpmg_fit_dir[strlen(cpmg_fit_dir)-1]!='/')
strcat(cpmg_fit_dir,"/");*/

strcpy(cpmg_fit_dir, "");
set_def(); /*Set initial guess for all necessary parameters namely R2, k's, dw's*/
if(parser());
exit(0);
}

int set_def()
{
int i,j,k,l,m;
init_kwords(); /* Include this part in the head_cp.h */

strcpy(prompt,d_prompt);

/* filling the structure par */
par.mod_=GM;
par.sim_=FALSE;
par.min_=FALSE;
par.clc_=TRUE;
par.clcu_=TRUE;
par.F_=par.P_lvl_=0;

```

```

par.df_=par.ndata_=par.nres_=par.nnucl_=par.ncoh_=par.nhsfrq_=0;
par.npark_=par.nparp_=par.nparR_=par.nparw_=0;
par.nprot_=0;

for(i=0;i<MAXPAR;i++)par.res_[i]=-1;
for(i=0;i<MAXPAR;i++)par.nucl_[i]=NONE;
for(i=0;i<MAXPAR;i++)par.hsfrq_[i]=0;

for(i=0;i<MAXPAR;i++)par.kmatrix[i]=0.;
for(i=0;i<MAXPAR;i++)par.pmatrix[i]=0.;
for(i=0;i<MAXDATA;i++)par.R2matrix[i]=0.;
for(i=0;i<MAXDATA;i++)par.dwmatrix[i]=0.;

/* two variables relevant to minimization */
mi.npar_=0;
mi.nfcn_=0;

return(0);
}

int parser()
{
char iline[MAXSTR], dummy[MAXSTR], fil[MAXSTR];
int i,j;
char c,*ctmp;
FILE *fn;

input=stdin;

while(1){
message(prompt);
if(NULL==(ctmp=fgets(iline,MAXSTR-1,input))){
if(input==stdin) exit(1);
else {fclose(fn);input=stdin;continue;}}
cutoff_comm(iline);
if(1>sscanf(iline,"%s",dummy)) continue;

/*if(iline[0]==' '){
strcat(strcat(strcpy(fil,"/bin/csh -c"),iline+1),"");
if(0>systemp(fil)){fprintf(stderr,"system call failed\n");}
continue;}*/

```

```

if(kword(iline)==RUN){
if(1!=sscanf(iline,"%*s%s",fil)){
fprintf(stderr,"use %s <macro file name>\n",find_kword(RUN));continue;}
if(NULL==(fn=fopen(fil,"r")))
{fprintf(stderr,"can't open file %s \n", fil);continue;}
else {input=fn;continue;}
}

switch(kword(iline)){
case EXIT: return(0); break;
case HELP:
    message("----- Help\n");
    if((-1)==help(iline)){fprintf(stdout,"command failed: retry\n");}
    break;
/*case SET: set_val(iline); break;*/
case READ: read_data(iline); break;
/*case WRITE: write_data(iline); break;*/
/*case DEL: del_dat(iline); break;*/
case MIN: minimize(iline); break;
/*case BACKUP: backup(iline); break;*/
/*case RESTORE: restore(iline); break;*/
case RUN: fprintf(stderr,"You cannot call macro from within macro\n"); break;
default: fprintf(stderr, "syntax error\n"); break;
}
}
}

int init_kwords()
{
strcpy(words[0].w,"q"); words[0].iw=EXIT;
strcpy(words[1].w,"exit"); words[1].iw=EXIT;
strcpy(words[2].w,"help"); words[2].iw=HELP;
strcpy(words[3].w,"set"); words[3].iw=SET;
strcpy(words[4].w,"read"); words[4].iw=READ;
strcpy(words[5].w,"del"); words[5].iw=DEL;
strcpy(words[6].w,"write"); words[6].iw=WRITE;
strcpy(words[7].w,"min"); words[7].iw=MIN;
strcpy(words[8].w,"backup"); words[8].iw=BACKUP;
strcpy(words[9].w,"restore"); words[9].iw=RESTORE;
strcpy(words[10].w,"run"); words[10].iw=RUN;
}

```

```
qsort((char*)words,N_WORDS,sizeof(struct dict),word_comp);
```

```
return(0);
}
```

```
int word_comp(s1,s2)
struct dict *s1,*s2;
{
int t,i;
for(i=0;i<=strlen(s1->w)&& i<=strlen(s2->w);i++)
{t=(toupper((int)(s1->w[i]))-toupper((int)(s2->w[i])));
if(t) return(t);
}
return(0);
}
```

```
int kword(s)
char *s;
{
int i;
char buff[MAXSTR];
if(1!=scanf(s,"%s",buff)) return(-1);
for(i=0;i<N_WORDS;i++)
{if(!strcmp(words[i].w, buff)) return(words[i].iw);}
return(-1);
}
```

```
char* find_kword(s)
int s;
{ int i;
for(i=0;i<N_WORDS;i++)
{if(words[i].iw==s) return(words[i].w);}
return(NULL);
}
```

```
char* cutoff_comm(s)
char *s;
{
if(NULL!=strchr(s,'#')) *strchr(s,'#')=0;
return(s);
}
```

```

}

int message(mess)
char *mess;
{fprintf(stdout,"%s",mess);return(0);}

int help(s)
char *s;
{;}

/* the same as malloc but make several trials */
char *rmalloc(size_t n)
{
int i;
char *p;
for(i=0;i<MAXALLOC_RETR;i++){
if(NULL!=(p=malloc(n))) break;
sleep(2);}
if(i==MAXALLOC_RETR) return(NULL);
else return(p);
}
/* This procedure is modified from Dmitry M. Korzhnev,
University of Toronto, Dept Medical genetics by Pulong Li
at UT Southwestern Medical Centre at Dallas Dec042005*/

/* Procedures (require compiled lapack and blas libraries):
R22s - Transverse relaxation rate for SQ measured using CPMG sequence
for the resonance of all sites in general case of 2-site exchange and
four edges, calculated as or  $-(1/\text{time\_T2}) \cdot \log(I1/I0)$ 
time_T2 - CPMG period length in second
I0      - Initial intensity of the resonance of all sites
I1      - intensity of the resonance of all site after CPMG period
Arguments:
*R2     - pointer to array R2[2] with R2 in states A, B, C and D
in second-1
*W     - pointer to array W[2] containing larmour frequencies of
states A, B, C and D in ppm
*P     - pointer to array P[2] containing populations of states
A, B, C and D in unitless
*K     - pointer to array K[2] containing the four microscopic
rate constants in second-1

```

```

    nrep    - number of d-180-d-d-180-d blocks in time_T2
    *R      - resiltng rate i.e. the calculated R2 to be returned
*/

#include <stdlib.h>
#include <stdio.h>
#include <complex.h>
#include <tgmath.h>
#include "math.h"
#include <values.h>
#include "head_cp.h"

void trans2m(double complex m[2][2]);
void conj2m(double complex m[2][2]);
void dot2m(double complex a[2][2], double complex b[2][2], double complex
r[2][2]);
void pow2m(double complex m[2][2], int n);

int RexCR(double *R2,double *W,double *P,double *K, double staField, double
tau,int nrep,double *R)
{
double complex dcp,psi,zeta,np,nm,Dp,Dm,dp,dm,zp,zm,mdp,mdm,mzp,mzm;
double tmp;
double R2A, R2B, DW, PA, KEX, DWH;

dcp=tau;
R2A=*R2;
R2B=*(R2+1);
DW=*W*staField;
PA=*P;
KEX=*K;
DWH=0;
psi=(R2A-(R2B-I*DWH)-PA*KEX+(1-PA)*KEX)*(R2A-(R2B-I*DWH)-
PA*KEX+(1-PA)*KEX)-DW*DW+4*PA*(1-PA)*KEX*KEX;
zeta=2*DW*(R2A-(R2B-I*DWH)-PA*KEX+(1-PA)*KEX);
np=sqrt(2)*dcp*sqrt(sqrt(psi*psi+zeta*zeta)+psi);
nm=sqrt(2)*dcp*sqrt(sqrt(psi*psi+zeta*zeta)-psi);
Dp=0.5*((psi+2*DW*DW)/sqrt(psi*psi+zeta*zeta)+1);
Dm=0.5*((psi+2*DW*DW)/sqrt(psi*psi+zeta*zeta)-1);
tmp=(double)creal(acosh(Dp*cosh(np)-Dm*cos(nm)));
if(tmp<0)tmp=-tmp;

```

```

*R=0.5*(R2A+R2B+KEX-1/(2*dcp)*tmp);
return(0);
}

int R22s(double *R2,double *W,double *P,double *K, double staField, double
tau,int nrep,double *R)
{
register int i, j, k;
register double kab, kba, time_T2, I0, I1;
double complex M1[2][2], V[2][2], IV[2][2], ID[2][2];
double complex res[2][2], tmp[2][2], tmp1[2][2];

/*variables for zgeev and zgetri */

char jovl='N', jovr='V';
int info, lwork=4, n=2, ipiv[2];
double rwork[4];
double complex A[2][2], vl[2][2], vr[2][2], w[2], work[4];

for(i=0;i<2;i++)
for(j=0;j<2;j++)M1[i][j]=0; /*Zeros M1[2][2]*/

memcpy(ID,M1,4*sizeof(double complex));
ID[0][0]=ID[1][1]=1; /*2x2 identity matrix*/

P[1]=1.0-(*P);
kab = *(K)*(P[1]/(*P+P[1]));
kba = *(K)*(*P)/(*P+P[1]);

A[0][0]=-(*R2)-kab; A[1][0]=kba;
A[0][1]=kab; A[1][1]=-I*(*W)*staField-(*R2)-kba;

zgeev_(&jovl, &jovr, &n, A, &n, w, vl, &n, vr, &n, work, &lwork, rwork, &info);
if(info!=0){fprintf(stderr,"zgeev erro%d/n",info);return(1);}

M1[0][0]=exp((*w*tau)); M1[1][1]=exp((*w+1)*tau);

memcpy(V,vr,4*sizeof(double complex));
trans2m(V);
memcpy(tmp,V,4*sizeof(double complex));
memcpy(IV,ID,4*sizeof(double complex));

```

```

zgesv_(&n,&n,tmp,&n,ipiv,IV,&n,&info);
if(info!=0){fprintf(stderr,"zgetri erro%d/n",info);return(1);}

dot2m(M1,IV,res);
dot2m(V,res,tmp);
memcpy(res,tmp,4*sizeof(double complex));
conj2m(tmp);
dot2m(tmp,res,tmp1);
memcpy(res,tmp1,4*sizeof(double complex));
conj2m(tmp1);
dot2m(tmp1,res,tmp);
memcpy(res,tmp,4*sizeof(double complex));

pow2m(res,nrep);
memcpy(tmp,res,4*sizeof(double complex));

I0=1;
I1=(double)creal(res[0][0]*P[0]+res[0][1]*P[1]+res[1][0]*P[0]+res[1][1]*P[1]);

time_T2=tau*4*nrep;

if(I1<=0||I1>=I0){fprintf(stderr,"Warning:I1 out of range:%g\n",I1);}
*R=-1/time_T2*log(I1/I0);

return(0);
}

int R22smajor(double *R2,double *W,double *P,double *K, double staField,
double tau,int nrep,double *R)
{
register int i, j, k;
register double kab, kba, time_T2, I0, I1;
double complex M1[2][2], V[2][2], IV[2][2], ID[2][2];
double complex res[2][2], tmp[2][2], tmp1[2][2];

/*variables for zgeev and zgetri */

char jovl='N', jovr='V';
int info, lwork=4, n=2, ipiv[2];
double rwork[4];
double complex A[2][2], vl[2][2], vr[2][2], w[2], work[4];

```

```

for(i=0;i<2;i++)
for(j=0;j<2;j++)M1[i][j]=0; /*Zeros M1[2][2]*/

memcpy(ID,M1,4*sizeof(double complex));
ID[0][0]=ID[1][1]=1; /*2x2 identity matrix*/

P[1]=1.0-(*P);
kab = *(K)*(P[1]/(*P+P[1]));
kba = *(K)*(*P)/(*P+P[1]);

A[0][0]=-I>(*W)*staField-(*R2)-kab; A[1][0]=kba;
A[0][1]=kab; A[1][1]=-I>(*W+1)*staField-(*R2+1)-kba;

zgeev_(&jovl, &jovr, &n, A, &n, w, vl, &n, vr, &n, work, &lwork, rwork, &info);
if(info!=0){fprintf(stderr,"zgeev erro%d/n",info);return(1);}

M1[0][0]=exp((*w*tau)); M1[1][1]=exp((*w+1)*tau);

memcpy(V,vr,4*sizeof(double complex));
trans2m(V);
memcpy(tmp,V,4*sizeof(double complex));
memcpy(IV,ID,4*sizeof(double complex));
zgesv_(&n,&n,tmp,&n,ipiv,IV,&n,&info);
if(info!=0){fprintf(stderr,"zgetri erro%d/n",info);return(1);}

dot2m(M1,IV,res);
dot2m(V,res,tmp);
memcpy(res,tmp,4*sizeof(double complex));
conj2m(tmp);
dot2m(tmp,res,tmp1);
memcpy(res,tmp1,4*sizeof(double complex));
conj2m(tmp1);
dot2m(tmp1,res,tmp);
memcpy(res,tmp,4*sizeof(double complex));

pow2m(res,nrep);
memcpy(tmp,res,4*sizeof(double complex));

I0=*(P);
I1=(double)creal(res[0][0]*P[0]+res[0][1]*P[1]);

```

```

time_T2=tau*4*nrep;

if(I1<=0||I1>=I0){fprintf(stderr,"Warning:I1 out of range:%g\n",I1);}
*R=-((1/time_T2)*log(I1/I0));

return(0);
}

void trans2m(double complex m[2][2])
{
register int i,j;
double complex a;
for(i=0;i<2;i++)
for(j=0;j<i;j++){a=m[i][j];m[i][j]=m[j][i];m[j][i]=a;}
}

void conj2m(double complex m[2][2])
{
register int i,j;
for(i=0;i<2;i++)
for(j=0;j<2;j++)m[i][j]=conj(m[i][j]);
}

void dot2m(double complex a[2][2], double complex b[2][2], double complex
r[2][2])
{
register int i,j,k;
for(i=0;i<2;i++)
for(j=0;j<2;j++){
r[i][j]=0;
for(k=0;k<2;k++){r[i][j]+=a[i][k]*b[k][j];}
}
}

void pow2m(double complex m[2][2], int n)
{
register int i,j;
double complex tmp[2][2], tmp1[2][2], tmp2[2][2];
memcpy(tmp,m,4*sizeof(double complex));
memcpy(tmp1,m,4*sizeof(double complex));

```

```

for (i=2;i<n;i=i*2){
dot2m(tmp,tmp1,tmp2);
memcpy(tmp,tmp2,2*sizeof(double complex));
memcpy(tmp1,tmp2,2*sizeof(double complex));}

for (j=i/2;j<n;j++){
dot2m(m,tmp,tmp1);
memcpy(tmp,tmp1,4*sizeof(double complex));}
memcpy(m,tmp,4*sizeof(double complex));
}
/* This procedure is modified from Dmitry M. Korzhnev,
University of Toronto, Dept Medical genetics by Pulong Li
at UT Southwestern Medical Centre at Dallas Dec042005*/

/* Procedures (require compiled lapack and blas libraries):
R23s - Transverse relaxation rate for SQ measured using CPMG sequence
for the resonance of all sites in general case of 3-site exchange and
four edges, calculated as  $-(1/\text{time\_T2}) \cdot \log(I1/I0)$ 
time_T2 - CPMG period length in second
I0 - Initial intensity of the resonance of all sites
I1 - intensity of the resonance of all site after CPMG period
Arguments:
*R2 - pointer to array R2[3] with R2 in states A, B, C and D
in second-1
*W - pointer to array W[3] containing larmour frequencies of
states A, B, C and D in ppm
*P - pointer to array P[3] containing populations of states
A, B, C and D in unitless
*K - pointer to array K[3] containing the four microscopic
rate constants in second-1
nrep - number of d-180-d-d-180-d blocks in time_T2
*R - resiltng rate i.e. the calculated R2 to be returned
*/

#include <stdlib.h>
#include <stdio.h>
#include <complex.h>
#include <tgmath.h>
#include "math.h"
#include <values.h>

```

```

#include "head_cp.h"

void trans3m(double complex m[3][3]);
void conj3m(double complex m[3][3]);
void dot3m(double complex a[3][3], double complex b[3][3], double complex
r[3][3]);
void pow3m(double complex m[3][3], int n);

int R23s(double *R2,double *W,double *P,double *K, double staField, double
tau,int nrep,double *R)
{
register int i, j, k;
register double kab, kba, kbc, kcb, kca, kac, time_T2, IO, I1, tau1;
double complex M1[3][3], V[3][3], IV[3][3], ID[3][3];
double complex res[3][3], tmp[3][3], tmp1[3][3];

/*variables for zgeev and zgetri */

char jovi='N', jovr='V';
int info, lwork=6, n=3, ipiv[3];
double rwork[6];
double complex A[3][3], vl[3][3], vr[3][3], w[3], work[6];

for(i=0;i<3;i++)
for(j=0;j<3;j++)M1[i][j]=0; /*Zeros M1[3][3]*/

memcpy(ID,M1,9*sizeof(double complex));
ID[0][0]=ID[1][1]=ID[2][2]=1; /*3x3 identity matrix*/

P[2]=1-*(P)-*(P+1);
kab = *(K)*(*(P+1)/(*P+*(P+1)));
kba = *(K)**(P)/(*P+*(P+1));
kbc = *(K+1)*(P[2]/(*(P+1)+P[2]));
kcb = *(K+1)**(P+1)/(*(P+1)+P[2]);
kca = *(K+2)**(P)/(P[2]+*(P));
kac = *(K+2)*(P[2]/(P[2]+*(P)));

A[0][0]=-I*(W)*staField-*(R2)-kab-kac; A[1][0]=kba; A[2][0]=kca;
A[0][1]=kab; A[1][1]=-I*(W+1)*staField-*(R2+1)-kba-kbc; A[2][1]= kcb;

```

```

A[0][2]=kac; A[1][2]=kbc; A[2][2]=-I*(*(W+2))*staField-(*(R2+1))-kcb-kca;
A[3][2]=kac;

zgeev_(&jovl, &jovr, &n, A, &n, w, vl, &n, vr, &n, work, &lwork, rwork, &info);
if(info!=0){fprintf(stderr,"zgeev erro%d/n",info);return(1);}

tau1=0.01/nrep;
M1[0][0]=exp((*w*tau1));
M1[1][1]=exp((*w+1)*tau1);M1[2][2]=exp((*w+2)*tau1);

memcpy(V,vr,9*sizeof(double complex));
trans3m(V);
memcpy(tmp,V,9*sizeof(double complex));
memcpy(IV,ID,9*sizeof(double complex));
zgesv_(&n,&n,tmp,&n,ipiv,IV,&n,&info);
if(info!=0){fprintf(stderr,"zgetri erro%d/n",info);return(1);}

dot3m(M1,IV,res);
dot3m(V,res,tmp);
memcpy(res,tmp,9*sizeof(double complex));
conj3m(tmp);
dot3m(tmp,res,tmp1);
memcpy(res,tmp1,9*sizeof(double complex));
conj3m(tmp1);
dot3m(tmp1,res,tmp);
memcpy(res,tmp,9*sizeof(double complex));

pow3m(res,nrep);
memcpy(tmp,res,9*sizeof(double complex));

I0=1;
I1=(double)creal(res[0][0]*P[0]+res[0][1]*P[1]+res[0][2]*P[2]+res[1][0]*P[0]+re
s[1][1]*P[1]
+res[1][2]*P[2]+res[2][0]*P[0]+res[2][1]*P[1]+res[2][2]*P[2]);

time_T2=tau*4*nrep;

if(I1<=0||I1>=I0){fprintf(stderr,"Warning:I1 out of range:%g\n",I1);}
*R=-((1/time_T2)*log(I1/I0));

return(0);

```

```

}

int R23smajor(double *R2,double *W,double *P,double *K, double staField,
double tau,int nrep,double *R)
{
register int i, j, k;
register double kab, kba, kbc, kcb, kca, kac, time_T2, I0, I1;
double complex M1[3][3], V[3][3], IV[3][3], ID[3][3];
double complex res[3][3], tmp[3][3], tmp1[3][3];

/*variables for zgeev and zgetri */

char jovl='N', jovr='V';
int info, lwork=6, n=3, ipiv[3];
double rwork[6];
double complex A[3][3], vl[3][3], vr[3][3], w[3], work[6];

for(i=0;i<3;i++)
for(j=0;j<3;j++)M1[i][j]=0; /*Zeros M1[3][3]*/

memcpy(ID,M1,9*sizeof(double complex));
ID[0][0]=ID[1][1]=ID[2][2]=1; /*3x3 identity matrix*/

P[2]=1-(*P)-(*P+1);
kab = *(K)*(*P+1)/(*P+*(P+1));
kba = *(K)*(*P)/(*P+*(P+1));
kbc = *(K+1)*P[2]/(*P+1)+P[2]);
kcb = *(K+1)*(*P+1)/(*P+1)+P[2]);
kca = *(K+2)*(*P)/(P[2]+*(P));
kac = *(K+2)*P[2]/(P[2]+*(P));

A[0][0]=-I*(*W)*staField-(*R2)-kab-kac; A[1][0]=kba; A[2][0]=kca;
A[0][1]=kab; A[1][1]=-I*(*W+1)*staField-(*R2+1)-kba-kbc; A[2][1]= kcb;
A[0][2]=kac; A[1][2]=kbc; A[2][2]=-I*(*W+2)*staField-(*R2+1)-kcb-kca;
A[3][2]=kac;

zgeev_(&jovl, &jovr, &n, A, &n, w, vl, &n, vr, &n, work, &lwork, rwork, &info);
if(info!=0){fprintf(stderr,"zgeev erro%d/n",info);return(1);}

```

```

M1[0][0]=exp((*w*tau));
M1[1][1]=exp((*w+1)*tau);M1[2][2]=exp((*w+2)*tau);

memcpy(V,vr,9*sizeof(double complex));
trans3m(V);
memcpy(tmp,V,9*sizeof(double complex));
memcpy(IV,ID,9*sizeof(double complex));
zgesv_(&n,&n,tmp,&n,ipiv,IV,&n,&info);
if(info!=0){fprintf(stderr,"zgetri erro%d/n",info);return(1);}

dot3m(M1,IV,res);
dot3m(V,res,tmp);
memcpy(res,tmp,9*sizeof(double complex));
conj3m(tmp);
dot3m(tmp,res,tmp1);
memcpy(res,tmp1,9*sizeof(double complex));
conj3m(tmp1);
dot3m(tmp1,res,tmp);
memcpy(res,tmp,9*sizeof(double complex));

pow3m(res,nrep);
memcpy(tmp,res,9*sizeof(double complex));

I0=*(P);
I1=(double)creal(res[0][0]*P[0]+res[0][1]*P[1]+res[0][2]*P[2]);

time_T2=tau*4*nrep;

if(I1<=0||I1>=I0){fprintf(stderr,"Warning:I1 out of range:%g\n",I1);}
*R=-1/time_T2*log(I1/I0);

return(0);
}

void trans3m(double complex m[3][3])
{
register int i,j;
double complex a;
for(i=0;i<3;i++)
for(j=0;j<i;j++){a=m[i][j];m[i][j]=m[j][i];m[j][i]=a;}
}

```

```

}

void conj3m(double complex m[3][3])
{
register int i,j;
for(i=0;i<3;i++)
for(j=0;j<3;j++)m[i][j]=conj(m[i][j]);
}

void dot3m(double complex a[3][3], double complex b[3][3], double complex
r[3][3])
{
register int i,j,k;
for(i=0;i<3;i++)
for(j=0;j<3;j++){
r[i][j]=0;
for(k=0;k<3;k++){r[i][j]+=a[i][k]*b[k][j];}
}
}

void pow3m(double complex m[3][3], int n)
{
register int i,j;
double complex tmp[3][3], tmp1[3][3], tmp2[3][3];
memcpy(tmp,m,9*sizeof(double complex));
memcpy(tmp1,m,9*sizeof(double complex));

for (i=2;i<n;i=i*2){
dot3m(tmp,tmp1,tmp2);
memcpy(tmp,tmp2,9*sizeof(double complex));
memcpy(tmp1,tmp2,9*sizeof(double complex));}

for (j=i/2;j<n;j++){
dot3m(m,tmp,tmp1);
memcpy(tmp,tmp1,9*sizeof(double complex));}
memcpy(m,tmp,9*sizeof(double complex));
}
/* This procedure is modified from Dmitry M. Korzhnev,
University of Toronto, Dept Medical genetics by Pulong Li
at UT Southwestern Medical Centre at Dallas Dec042005*/

```

```

/* Procedures (require compiled lapack and blas libraries):
R24s - Transverse relaxation rate for SQ measured using CPMG sequence
for the resonance of all sites in general case of 4-site exchange and
four edges, calculated as or  $-(1/\text{time\_T2}) \cdot \log(I1/I0)$ 
    time_T2 - CPMG period length in second
    I0      - Initial intensity of the resonance of all sites
    I1      - intensity of the resonance of all site after CPMG period
Arguments:
    *R2     - pointer to array R2[4] with R2 in states A, B, C and D
             in second-1
    *W      - pointer to array W[4] containing larmour frequencies of
             states A, B, C and D in ppm
    *P      - pointer to array P[4] containing populations of states
             A, B, C and D in unitless
    *K      - pointer to array K[4] containing the four microscopic
             rate constants in second-1
    nrep    - number of d-180-d-d-180-d blocks in time_T2
    *R      - resliting rate i.e. the calculated R2 to be returned
*/

#include <stdlib.h>
#include <stdio.h>
#include <complex.h>
#include <tgmath.h>
#include "math.h"
#include <values.h>
#include "head_cp.h"

void trans4m(double complex m[4][4]);
void conj4m(double complex m[4][4]);
void dot4m(double complex a[4][4], double complex b[4][4], double complex
r[4][4]);
void pow4m(double complex m[4][4], int n);

int R24s(double *R2, double *W, double *P, double *K, double staField, double
tau, int nrep, double *R)
{
    register int i, j, k;
    register double kab, kba, kbc, kcb, kcd, kdc, kda, kad, time_T2, I0, I1, tau1;
    double complex M1[4][4], V[4][4], IV[4][4], ID[4][4];
    double complex res[4][4], tmp[4][4], tmp1[4][4];

```

```

/*variables for zgeev and zgetri */

char jowl='N', jovr='V';
int info, lwork=8,n=4, ipiv[4];
double rwork[8];
double complex A[4][4], vl[4][4], vr[4][4], w[4], work[8];

for(i=0;i<4;i++)
for(j=0;j<4;j++)M1[i][j]=0; /*Zeros M1[4][4]*/

memcpy(ID,M1,16*sizeof(double complex));
ID[0][0]=ID[1][1]=ID[2][2]=ID[3][3]=1; /*4x4 identity matrix*/

kab = exp(*(K))*(*(P+1)/(*P+*(P+1)));
kba = exp(*(K))**(P)/(*P+*(P+1));
kbc = exp(*(K+1))**(P+2)/(*P+1+*(P+2));
kcb = exp(*(K+1))**(P+1)/(*P+1+*(P+2));
kcd = exp(*(K+2))**(P+3)/(*P+2+*(P+3));
kdc = exp(*(K+2))**(P+2)/(*P+2+*(P+3));
kda = exp(*(K+3))**(P)/(*P+*(P+3));
kad = exp(*(K+3))**(P+3)/(*P+*(P+3));

A[0][0]=-I*(*W)*staField-(*(R2))-kab-kad; A[1][0]=kba; A[2][0]=0; A[3][0]=kda;
A[0][1]=kab; A[1][1]=-I*(*W+1)*staField-(*(R2))-kba-kbc; A[2][1]= kcb;
A[3][1]=0;
A[0][2]=0; A[1][2]=kbc; A[2][2]=-I*(*W+2)*staField-(*(R2))-kcb-kcd;
A[3][2]=kdc;
A[0][3]=kad; A[1][3]=0; A[2][3]=kcd; A[3][3]=-I*(*W+3)*staField-(*(R2))-
kda-kdc;

zgeev_(&jowl, &jovr, &n, A, &n, w, vl, &n, vr, &n, work, &lwork, rwork, &info);
if(info!=0){fprintf(stderr,"zgeev erro%d/n",info);return(1);}

tau1=0.04/(4*nrep);

M1[0][0]=exp(*(w*tau1)); M1[1][1]=exp(*(w+1)*tau1);
M1[2][2]=exp(*(w+2)*tau1); M1[3][3]=exp(*(w+3)*tau1);

memcpy(V,vr,16*sizeof(double complex));
trans4m(V);

```

```

memcpy(tmp,V,16*sizeof(double complex));
memcpy(IV,ID,16*sizeof(double complex));
zgesv_(&n,&n,tmp,&n,ipiv,IV,&n,&info);
if(info!=0){fprintf(stderr,"zgetri erro%d/n",info);return(1);}

dot4m(M1,IV,res);
dot4m(V,res,tmp);
memcpy(res,tmp,16*sizeof(double complex));
conj4m(tmp);
dot4m(tmp,res,tmp1);
memcpy(res,tmp1,16*sizeof(double complex));
conj4m(tmp1);
dot4m(tmp1,res,tmp);
memcpy(res,tmp,16*sizeof(double complex));

pow4m(res,nrep);
memcpy(tmp,res,16*sizeof(double complex));

I0=1;
I1=(double)creal(res[0][0]*P[0]+res[0][1]*P[1]+res[0][2]*P[2]+res[0][3]*P[3]
+res[1][0]*P[0]+res[1][1]*P[1]+res[1][2]*P[2]+res[1][3]*P[3]
+res[2][0]*P[0]+res[2][1]*P[1]+res[2][2]*P[2]+res[2][3]*P[3]
+res[3][0]*P[0]+res[3][1]*P[1]+res[3][2]*P[2]+res[3][3]*P[3]);

/*time_T2=tau*4*nrep;*/

/*if(I1<=0||I1>=I0){fprintf(stderr,"Warning:I1 out of range:%g\n",I1);}*/
*R=-((1/0.04)*log(I1/I0));

return(0);
}

int R24smajor(double *R2,double *W,double *P,double *K, double staField,
double tau,int nrep,double *R)
{
register int i, j, k;
register double kab, kba, kbc, kcb, kcd, kdc, kda, kad, time_T2, I0, I1;
double complex M1[4][4], V[4][4], IV[4][4], ID[4][4];
double complex res[4][4], tmp[4][4], tmp1[4][4];

/*variables for zgeev and zgetri */

```

```

char jowl='N', jovr='V';
int info, lwork=8,n=4, ipiv[4];
double rwork[8];
double complex A[4][4], vl[4][4], vr[4][4], w[4], work[8];

for(i=0;i<4;i++)
for(j=0;j<4;j++)M1[i][j]=0; /*Zeros M1[4][4]*/

memcpy(ID,M1,16*sizeof(double complex));
ID[0][0]=ID[1][1]=ID[2][2]=ID[3][3]=1; /*4x4 identity matrix*/

P[3]=1-(*P)-(*P+1)-(*P+2);
kab = *(K)*(*P+1)/(*P+*(P+1));
kba = *(K)*(*P)/(*P+*(P+1));
kbc = *(K+1)*(*P+2)/(*P+1)+*(P+2));
kcb = *(K+1)*(*P+1)/(*P+1)+*(P+2));
kcd = *(K+2)*P[3]/(*P+2)+P[3]);
kdc = *(K+2)*(*P+2)/(*P+2)+P[3]);
kda = *(K+3)*(*P)/(*P)+P[3]);
kad = *(K+3)*P[3]/(*P)+P[3]);

A[0][0]=-I*(*W)*staField-(*R2)-kab-kad; A[1][0]=kba; A[2][0]=0; A[3][0]=kda;
A[0][1]=kab; A[1][1]=-I*(*W+1))*staField-(*R2+1))-kba-kbc; A[2][1]= kcb;
A[3][1]=0;
A[0][2]=0; A[1][2]=kbc; A[2][2]=-I*(*W+2))*staField-(*R2+1))-kcb-kcd;
A[3][2]=kdc;
A[0][3]=kad; A[1][3]=0; A[2][3]=kcd; A[3][3]=-I*(*W+3))*staField-(*R2)-kda-
kdc;

zgeev_(&jowl, &jovr, &n, A, &n, w, vl, &n, vr, &n, work, &lwork, rwork, &info);
if(info!=0){fprintf(stderr,"zgeev erro%d/n",info);return(1);}

M1[0][0]=exp((*w*tau)); M1[1][1]=exp((*w+1)*tau);
M1[2][2]=exp((*w+2)*tau); M1[3][3]=exp((*w+3)*tau);

memcpy(V,vr,16*sizeof(double complex));
trans4m(V);
memcpy(tmp,V,16*sizeof(double complex));
memcpy(IV,ID,16*sizeof(double complex));
zgesv_(&n,&n,tmp,&n,ipiv,IV,&n,&info);

```

```

if(info!=0){fprintf(stderr,"zgetri erro%d/n",info);return(1);}

dot4m(M1,IV,res);
dot4m(V,res,tmp);
memcpy(res,tmp,16*sizeof(double complex));
conj4m(tmp);
dot4m(tmp,res,tmp1);
memcpy(res,tmp1,16*sizeof(double complex));
conj4m(tmp1);
dot4m(tmp1,res,tmp);
memcpy(res,tmp,16*sizeof(double complex));

pow4m(res,nrep);
memcpy(tmp,res,16*sizeof(double complex));

I0=*(P);

I1=(double)creal(res[0][0]*P[0]+res[0][1]*P[1]+res[0][2]*P[2]+res[0][3]*P[3]);

time_T2=tau*4*nrep;

if(I1<=0||I1>=I0){fprintf(stderr,"Warning:I1 out of range:%g\n",I1);}
*R=-1/time_T2*log(I1/I0);

return(0);
}

void trans4m(double complex m[4][4])
{
register int i,j;
double complex a;
for(i=0;i<4;i++)
for(j=0;j<i;j++){a=m[i][j];m[i][j]=m[j][i];m[j][i]=a;}
}

void conj4m(double complex m[4][4])
{
register int i,j;
for(i=0;i<4;i++)
for(j=0;j<4;j++)m[i][j]=conj(m[i][j]);
}

```

```

void dot4m(double complex a[4][4], double complex b[4][4], double complex
r[4][4])
{
register int i,j,k;
for(i=0;i<4;i++)
for(j=0;j<4;j++){
r[i][j]=0;
for(k=0;k<4;k++){r[i][j]+=a[i][k]*b[k][j];}
}
}

```

```

void pow4m(double complex m[4][4], int n)
{
register int i,j;
double complex tmp[4][4], tmp1[4][4], tmp2[4][4];
memcpy(tmp,m,16*sizeof(double complex));
memcpy(tmp1,m,16*sizeof(double complex));

for (i=2;i<n;i=i*2){
dot4m(tmp,tmp1,tmp2);
memcpy(tmp,tmp2,16*sizeof(double complex));
memcpy(tmp1,tmp2,16*sizeof(double complex));}

for (j=i/2;j<n;j++){
dot4m(m,tmp,tmp1);
memcpy(tmp,tmp1,16*sizeof(double complex));}
memcpy(m,tmp,16*sizeof(double complex));
}

```

Appendix 12 List of Fortran modules used in Appendix 11

Information about all of the following Fortran modules except sa.f can be found on this website: <http://www.netlib.org/slatec/src/>

d1mach.f
dckder.f
dcov.f
denorm.f
dfdjc3.f
dgamln.f
dmpar.f
dnls1.f
dqags.f
dqagse.f
dqelg.f
dqk21.f
dqpsrt.f
dqrfac.f
dqrslv.f
dwupdt.f
fdump.f
i1mach.f
j4save.f
xercnt.f
xerhlt.f
xermsg.f
xerprn.f
xersve.f
xgetua.f

Appendix 13 Makefile script I

```
#!/bin/csh -f

setenv CPMG_FIT cpmg_fit_linux8      # Program name
setenv ETC_LIB libetc.a              # Library with SLATECK subroutines
setenv CC gcc                        # C compiler, should be gcc with complex
number support
setenv FC g77                        # F77 compiler
setenv CFLAGS "-DLINUX -O2"         # C Compillation options
setenv FFLAGS "-ftypeless-boz -O2"  # F77 Compillation options
set LAPACK=('/usr/lib/liblapack.a')  # Path to LAPACK
set BLAS=('/usr/lib/libblas.a')     # Path to BLAS
setenv LIB_LAPACK "$LAPACK"
setenv LIB_BLAS "$BLAS"

echo "Removing object files and libraries..."
rm -f *.o */*.o
rm -f *.a

echo "compile F77 routines from SLATEC lib..."
cd F
make
mv $ETC_LIB ..
cd ..

echo ""
echo "make $CPMG_FIT ..."
make

echo "Removing object files and libraries..."
rm -f *.o */*.o
rm -f *.a
echo "done ..."
```

Appendix 14 Makefile script II

```
# makefile for cpmg_fit

OBJECTS = pars.o read_data_parameter.o R2_4_sites.o R2_3_sites.o
R2_2_sites.o minimization.o quantil.o

.c.o:
    $(CC) $(CFLAGS) -c $<
.f.o:
    $(FC) $(FFLAGS) -c $<

$(CPMG_FIT) : $(OBJECTS)
    $(FC) $(CFLAGS) $(OBJECTS) -o $(CPMG_FIT) $(ETC_LIB)
$(LIB_LAPACK) $(LIB_BLAS) -lm

$(OBJECTS) : head_cp.h slatec.h
```