AN ENERGY LANDSCAPING APPROACH TO THE PROTEIN FOLDING PROBLEM

A Thesis
Presented to
The Academic Faculty

by

Temsiri Sapsaman

In Partial Fulfillment
of the Requirements for the Degree
Doctor of Philosophy in the
The George W. Woodruff School of Mechanical Engineering

Georgia Institute of Technology
December 2009

AN ENERGY LANDSCAPING APPROACH TO THE PROTEIN FOLDING PROBLEM

Approved by:

Professor Harvey Lipkin,
Committee Chair
The George W. Woodruff School of
Mechanical Engineering
Georgia Institute of Technology

Professor Stephen C. Harvey
School of Biology
Georgia Institute of Technology

Professor Nader Sadegh
The George W. Woodruff School of
Mechanical Engineering
Georgia Institute of Technology

Professor Joel S. Sokol
The School of Industrial System and
Engineering
Georgia Institute of Technology

Professor Michael J. Leamy
The George W. Woodruff School of
Mechanical Engineering
Georgia Institute of Technology

Date Approved: 13 November 2009

To my grandmother,

***Puangtip Tiamtrakul**,*

and my parents,

***Pojana and Wut Sapsaman**.*

ACKNOWLEDGEMENTS

I would like express my sincere gratitude to everyone supporting me in completing this thesis.

My foremost gratitude goes to my adviser, Dr. Harvey Lipkin, a very knowledgeable individual who likes to see students succeed and from whom I have learned so much. Thank you for always leaving your door open.

I also would like to show my appreciation to Dr. Nader Sadegh, Dr. Michael Leamy, Dr. Steve Harvey, and Dr. Joel Sokol for serving as committee members, to Dr. Jeffrey Donnell for proofreading, and to Jane Chisholm for improving my writing skills.

My heartfelt thanks go to my officemate, my fellow Thai student, "Jiggy" Jomkwun Munnae, for the best advice, constant encouragement, and always being my best friend. My special thanks go to Thom Sokol for editing the thesis and giving constructive criticism, for constant love and support, and for always believing in me.

My deepest appreciation goes to the George W. Woodruff School of Mechanical Engineering for financial support and the opportunities to teach. I also would like to thank Mechanical Engineering Graduate office staff for helping with paperwork and giving emotional support.

Last but certainly not least, I would like to thank my family and Thom's family for encouragement and friends from Woodruff School Graduate Women and Thai Student Organization for great company.

Temsiri Sapsaman

TABLE OF CONTENTS

LIST OF TABLES

LIST OF FIGURES

LIST OF SYMBOLS OR ABBREVIATIONS

AXEL Adaptive Exponential Energy Landscaping.

BFGS Broyden-Fletcher-Goldfarb-Shanno algorithm.

BFGS-XEL Broyden-Fletcher-Goldfarb-Shanno algorithm with Exponential Energy Landscaping.

Broyden-XEL Broyden's Method with Exponential Energy Landscaping.

Newton-XEL Newton's method with Exponential Energy Landscaping.

QNA Quasi-Newton Algorithm.

QNA-XEL Quasi-Newton Algorithm with Exponential Energy Landscaping.

XEL Exponential Energy Landscaping.

α Line search gain.

α^* Line search gain that implicitly calculates β.

β $\beta = n\,|E|^{n-1}$.

β^k Redefined β or β with n_k: $\beta_k^k = n_k\,|E_k|^{n_k-1}$ and $\beta_{k-1}^k = n_k\,|E_{k-1}|^{n_k-1}$.

δ Bounding constant.

κ^* Hessian matrix of the adaptively modified energy landscape.

$\hat{\kappa}^*$ Estimated Hessian matrix of the adaptively modified energy landscape.

λ^Θ Scalar multiplier for varying-n Newton-XEL step: $\hat{h}_\theta^* = \lambda^\Theta \hat{h}$.

ρ Error residue of Newton's method.

ρ^* Error residue of Newton-XEL.

τ Torque vector.

τ^* Torque vector of the modified energy landscape: $\tau^* = \beta\tau$.

$\Delta\tau$ $\Delta\tau_{k-1} = \tau_k - \tau_{k-1}$.

$\Delta\tau^*$ $\Delta\tau_{k-1}^* = \tau_k^* - \tau_{k-1}^*$.

$\Delta\tau^{*\,k}$ Redefined $\Delta\tau^*$ or $\Delta\tau^*$ with n_k: $\Delta\tau^{*\,k}_{k-1} = \beta_k^k\tau_k - \beta_{k-1}^k\tau_{k-1}$.

θ A vector variable of a cost function. For protein conformation prediction, the dihedral angle vector.

Θ	A composite vector variable of the dihedral angle vector and the exponent of the XEL.
$AveSI$	Average score improvement.
$\%\Delta AveSI$	The percent difference of the average score improvement.
$AveIt$	Average iteration.
$\%\Delta AveIt$	The percent difference of the average iteration.
e	Efficiency.
$\%\Delta e$	The percent difference of the efficiency.
E	Energy function.
E^*	Modified energy function.
E_{lowest}	Lowest energy.
$\%\Delta E_{\mathsf{lowest}}$	The percent difference of the lowest energy.
f	Cost function.
h	A step for the dihedral angles.
$\hat{h}$	An optimization direction. For protein conformation prediction, a change in the dihedral angle vector.
h^*	A step for the dihedral angles of the modified energy function.
$\hat{h}^*$	An optimization direction of the XEL algorithm. For protein conformation prediction, a change in the dihedral angle vector.
$\hat{h}^*_\theta$	A step for θ.
$\hat{h}^*_\Theta$	A step for Θ.
$\hat{h}^*_n$	A step for n.
$\hat{H}$	The estimation of the Hessian matrix inverse K^{-1}.
$\hat{H}^*$	The estimation of the modified Hessian matrix inverse $K^{*\,-1}$.
K	The second derivative or a Hessian matrix. For protein conformation prediction, the stiffness matrix.
$\hat{K}$	The estimation of the second derivative or a Hessian matrix. Hessian matrix of the QNA method. For protein conformation prediction, the estimated stiffness matrix.

K^*	The second derivative or the Hessian matrix of the modified energy landscape.
$\hat{K^*}$	The estimated second derivative or the estimated Hessian matrix of the modified energy landscape.
n	The exponent of the XEL.
SI	Score improvement.

SUMMARY

The function of a protein is largely dictated by its natural shape called the "native conformation." Since the native conformation and the global minimum energy configuration highly correlate, predicting this conformation is a global optimization known as the "protein folding problem." It is computationally intensive due to the high-dimensional and complex energy landscape. Typical conformation algorithms combine a probabilistic search with analytical optimization [56]. The analytical portion typically takes longer than the probabilistic part since more function evaluations are required, which are algorithm bottlenecks.

To reduce the computational cost, this research studies the effects of exponential energy landscaping (XEL) on three analytical optimization algorithms: Newton's method, a quasi-Newton algorithm (QNA), and the Broyden-Fletcher-Goldfarb-Shanno (BFGS) algorithm. The XEL changes the heights and the depths of the extrema but keeps their location the same, which eliminates the troublesome process of remapping minima onto the original landscape. The Newton-XEL is found to have a similar convergence property as Newton's method by showing that their error residues are of the same order. Found by observation, stability and convergence are improved when the error residue is bounded. While XEL is found to have no effect on the similarity of resulting configurations to the native conformation, results show that the XEL can improve the speed in terms of average iterations in the QNA by 47% and in the BFGS by 41%. In terms of the average score improvement, which indicates how the energy of the resulting configuration is compared to that of the initial configuration, the XEL can improve the quality of resulting configurations in the QNA by 12% and in the BFGS by 10%. Since both results were not achieved simultaneously, the

adaptive exponential energy landscaping (AXEL) is developed. The results lead to the conclusion that a trade-off between quality and speed must be considered when XEL is implemented. To improve speed by 15% to 47% and efficiency by 13% to 75%, XEL with n within 2^{-9}–2^{-5} should be used and to improve quality by 4% to 7%, AXEL with Scheme E that keeps the error residue bounded should be used.

CHAPTER I

INTRODUCTION

1.1 Motivation

The chemicals that play the primary role in performing, catalyzing, or supporting the biological functions of organisms are proteins [35]. As researchers have known for many years, the functionality of proteins is largely determined from their structural conformations. For example, in the human immune system, an antibody has a shape that only a certain antigen can fit. In drug design, the functions of new proteins are traditionally determined by conducting numerous experimental trials. However, if the protein conformation is known in advance the number of experimental trials can be reduced by using computer simulation to determine substances whose shapes may potentially react to the protein. This process could increase the success rate of pharmaceutical trials.

Since the native conformation and the global minimum energy configuration highly correlate, predicting the protein conformation is a global minimization problem. It is a computationally intensive task due to the high-dimensional and the complex energy landscape. Because on average a protein molecule has several thousands atoms, it can have several hundreds degrees of freedom even with a reduced model. This creates the large energy landscape and an extremely intensive search is generally required. Additionally, a very rough energy landscape also makes the minimization difficult because of more local minima. With atomic energy associated to each atom pair, the energy of the protein drastically varies as the protein configuration changes. In [11] the protein conformation prediction process takes about 150 CPU days for each small protein.

1.2 Protein Structure

A protein molecule is a chain of amino acid units known as residues [36]. The sequence of amino acids is called a protein sequence. Twenty types of amino acids are identified by their side chains, indicated by R_i in Figure 1.1, which illustrates a series of four residues. These side chains connect to alpha-carbon atoms of the amino acids and have zero to four degrees of rotational freedom. The configurations of the side chains depend on the values of their rotation angles, known as the Rotamer angles.

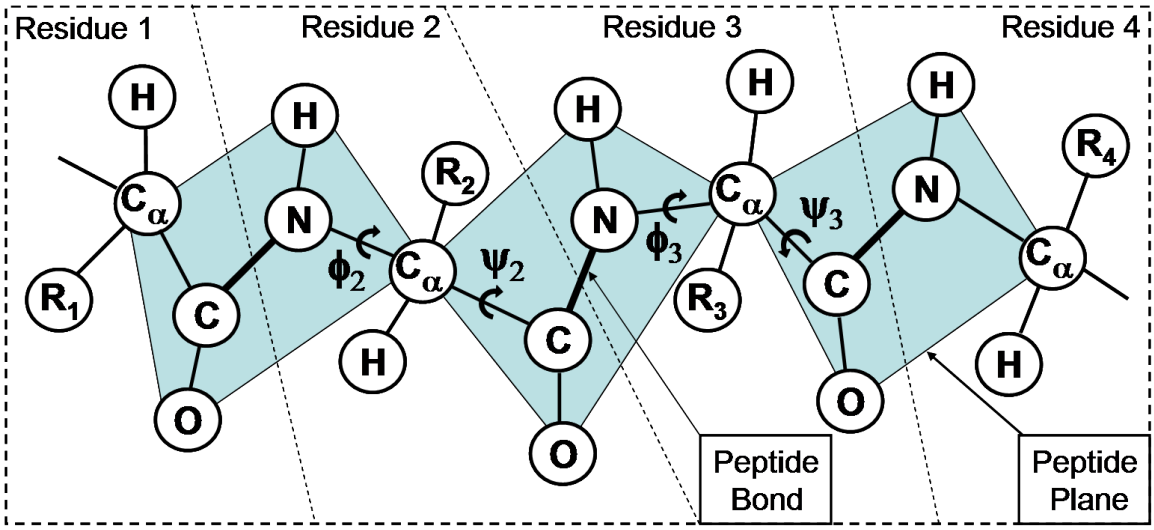

Figure 1.1: A series of three peptide planes.

Between the pairs of residues strong chemical bonds are formed between nitrogen and carbon atoms called "peptide bonds." Each of these bonds creates a relatively rigid plane between two residues, called a "peptide plane." The peptide plane consists of alpha-carbon (C_α), carbon (C), and oxygen (O) atoms from one residue and nitrogen (N), hydrogen (H), and alpha-carbon atoms from another. Between each pair of peptide planes, the molecule can rotate along the N–C_α and C_α–C bonds. The values of these rotation angles, ϕ and ψ, called "dihedral angles," largely determine the shape of a protein.

1.3 Protein Folding Process

Protein folding is the process by which a protein chain transforms into a stable conformation [34, 35, 36]. The process often starts when a protein chain is stretched out and non-functional, in a "de-natured" state. The protein is unstable and will naturally fold to obtain the stable "native state" due to atomic forces. In the native state the protein chain is naturally clustered and has minimum potential energy. Some proteins can have more than one native conformation in different environments or solvents. However, when the environmental properties are constant the protein always folds into one native conformation. Interestingly, the folding path by which this conformation is reached does not have a monotonically-decreasing potential energy. Instead, the protein often encounters energetic barriers before it reaches the conformation that has minimum potential energy. In overcoming these energy barriers the protein has to fold into a higher energy configuration and move against atomic forces. This motion is a result of non-zero kinetic energy occurring as a consequence of a decrease in potential energy. Researchers believe that these energy barriers separate the native state from the de-natured state and prevent the protein from unfolding in response to small disturbances [28].

1.4 Methods of Protein Conformation Prediction

Also known as the protein folding problem, the protein conformation prediction problem is the challenge of determining the native conformation of a new protein. This problem is approached with several methods of conformation prediction, including molecular dynamics simulation and optimization.

1.4.1 Molecular Dynamics

The effort to determine the conformation of the native state has involved the use of molecular dynamics (MD) simulation [14] which simulates the motion of a molecule.

The time step of an accurate simulation must be very small, typically on the order of femtoseconds (10^{-15} s). In contrast, the folding process takes a much longer time, typically on the order of milliseconds or seconds. Therefore, the MD simulations are computationally expensive [60]. This technique can locate the final conformation for a very small molecule but it is inefficient for larger proteins [7]. Since the smallest amino acid has 10 atoms and roughly 25 degrees of freedom that allow bond stretching, angle bending, and torsional motions, an atomic-level model of a small protein with 100 residues approximately contains at least 2,500 degrees of freedom. As a result, dynamic simulation is impractical because of time constraints. Many studies address this problem through reducing the degrees of freedom of the protein molecule model [53, 54]. Since the motions of atoms on a peptide plane relative to one another are very small, one way to significantly reduce the degrees of freedom is to treat the peptide planes as rigid bodies. This model only allows motions through changes in the dihedral and the Rotamer angles. Because each residue has two dihedral angles, the main chain of an n-residue protein model has $2n$ degrees of freedom. Although this reduction of computational costs could be an order of magnitude, the MD simulation is still not practical for solving the folding problem of large proteins.

Another inaccuracy arises from the empirical multidimensional energy function, also called a force field model, which is used to calculate atomic energies and forces. Expressing the potential energies of a protein molecule as a function of the molecular structure, the force field model yields a reduction of the computational costs compared to a quantum mechanics approach but is less accurate [44].

1.4.2 Robot Model

In addition to computational cost reduction, another advantage of modeling a molecule as a chain of rigid bodies is that robotic kinematics, dynamics, and controls methods can be applied if the protein molecule is considered as a nanomanipulator

[3, 34, 36, 61]. A serial manipulator has revolute-joint angles and rigid links corresponding to dihedral angles and peptide planes respectively. An example of robotic studies is the work by Sharma *et al.* [61] who identify the protein reachable workspace using protein kinematics. However, an application of this workspace analysis may not be available in the near future because independently manipulating each individual joint of the molecule is not yet possible. Other studies [29, 30, 71, 75] present more sophisticated models that reduce the degrees of freedom further by identifying the rigid parts of the known protein chain due to constraints such as covalent and hydrogen bonds and only allow motions at the flexible parts of the chain. These techniques are helpful for the study of the operational protein motion such as binding, but they do not help the study of the folding process from the de-natured state because the constraints are unknown.

1.4.3 Optimization

Since at the native state proteins are in a minimum energy configuration, most research on protein conformation prediction uses optimization algorithms that search the conformational space to locate the conformation with the minimum energy. The current statistics on the PDB data show that on average a protein has 587 residues and the maximum protein size is 18,540 residues [1]. In this high-dimensional space, an extremely intensive search is therefore required. Additionally, the difficulties of the search arise due to the imprecise and highly complex force field model.

Although there are numerous algorithms for the protein-folding problem, all are variations of an analytical or statistical optimization algorithm, or a combination of both. The most popular statistical optimization algorithm used in the protein-folding problem is likely the Monte Carlo (MC) method [44], which is the basis of most probabilistic searches [56]. MC simulation generates a collection of statistical configurations (or decoys) by adding random changes to the current configuration

and accepting or rejecting them under the Metropolis criterion [44]. For a large protein, the MC algorithm can be inefficient because most generated configurations will be rejected. Therefore, a combination of MC and other optimization techniques is generally used in practice. For example, the Rosetta software [56], which is discussed in more detail in Chapter 5, randomly chooses pre-generated changes that are more energetically suitable for parts of the protein sequence to increase the acceptance rate.

One example of an analytical optimization algorithm, the method of steepest descent (SD), that is used in conjunction with the robotic model of the protein molecule is the work by Kazerounian *et al.* [34]. Noting that the minimum-energy conformation has zero torques about the dihedral angle axes, they developed the successive kineto-static compliance (SKSC) method which is a simple control law that alters the protein configuration until zero torques are reached. The algorithm is capable of finding a configuration with joint torques close to zero; however, this configuration does not guarantee the native state but rather a local minimum at which the torques are also zero. Due to the complexity of the energy landscape, this criterion is not sufficient for finding the global minimum energy. Their extended work [9] modifies the energy landscape by using segmental manipulation and progressive side-chain scaling. Although the new algorithm yields better results, the final configurations are still not in the native configuration.

1.5 Bottleneck

Analytical optimization algorithms can locate a local solution faster than probabilistic optimization algorithms, but the analytical optimization algorithms by themselves are unsuccessful in a global search because of the high-dimensional and complex energy landscape. Therefore, most protein conformation prediction algorithms combine a probabilistic search and analytical optimization algorithms [56]. The combined algorithms probabilistically generate several initial configurations (decoys) for local

searching so to cover more areas in the energy landscape. Each local search progresses by slightly changing the shape of each decoy to yield an initial guess and finds a local minimum near the perturbed configuration. Then the resulting configuration is perturbed again to yield another initial guess. This cycle of perturbation and minimization is usually repeated hundreds of times and is called "refinement". There are usually several rounds of refinement and only most promising decoys are refined in further rounds.

Because more function evaluations are required, the analytical optimization algorithms usually take longer than the probabilistic portion and they become bottlenecks in the protein conformation prediction process. The efficiency of the analytical minimization becomes more problematic due to the following reasons: 1) high degrees of freedom in the protein molecule yields a high-dimensional minimization problem which is difficult and time consuming, 2) the refinement process involves several hundreds cycles of minimizations, and 3) to cover a large search area several thousands of decoys are needed even for a small protein and these decoys go through refinement. As the protein gets larger this problem gets magnified. To put it into perspective, in a high-resolution structure prediction [11] 20,000 to 30,000 decoys are generated for each small protein with 85 or fewer residues and the entire process for each protein takes 150 CPU days. To reduce the computational cost, the efficiency of the analytical algorithms needs to be addressed.

1.6 Contribution

This work investigates the effects of the method of energy landscaping on three analytical algorithms: Newton's method, a quasi-Newton algorithm (QNA) and the Broyden-Fletcher-Goldfarb-Shanno (BFGS) algorithm. The investigated landscaping, called the method of Exponential Energy Landscaping (XEL) [58], changes the

heights and the depths of the extrema but keeps their location the same, which eliminates the troublesome process of remapping minima onto the original landscape. The simulation results show that in general the method of XEL uniformly affects the performance of all investigated algorithms. Since different degrees of the energy landscape modification yield different performances, an adaptively modified energy landscape (AXEL) scheme has been developed to dynamically modify the energy landscape based on the current performance of the algorithm. For minimization on some of the AXEL, varying-n XEL algorithms have been developed, taking into account the effect of the changing energy landscape. The effects of AXEL schemes on the XEL and the varying-n XEL variations of the three above algorithms are investigated.

The goal of this work is to improve the speed and quality of the analytical optimization algorithms using an exponential energy landscaping. With this energy landscaping, the computational cost can be reduced by up to 47% and a protein conformation with up to 30% lower energy can be located. Although both results were not achieved simultaneously, the exponential energy landscaping can speed up the prediction process in balancing quality of resulting configurations and computational cost.

1.7 Organization

This thesis is organized into six chapters. Chapter I discusses the motivation of the research, introduces the protein structure and the protein folding problem, and briefly reviews the methods of protein conformation prediction. Chapter II presents the literature review of several aspects in the protein conformation prediction problem, such as problem formulation and algorithms. Chapter III discusses the theoretical background for two analytical optimization algorithms, Newton's method and quasi-Newton methods, and their applications to the protein folding problem. Chapter IV presents and discusses the method of exponential energy landscaping and the

adaptively modified energy landscape. Chapter V presents simulation results, observations, and discussions. Lastly, Chapter VI presents a conclusion and discusses future work.

CHAPTER II

LITERATURE REVIEW

Protein analysis can be basically categorized into two problems. The first is to identify the sequence of amino acids in a protein when the conformation is known [82], and the second is to identify the conformation of a protein when the sequence is known. Both conformation and sequence predictions are studied in single proteins as well as in protein-protein interactions.

The methods of predicting protein conformation are classified into three groups: comparative modeling, fold recognition, and *ab initio* methods. Both comparative modeling and fold recognition methods assume that the conformation of an unknown protein is comparable to that of a known protein with a similar sequence. Utilizing the protein database (PDB), which contains comprehensive data of studied proteins, both methods compare the sequence of the unknown to that of known proteins. Because the comparative modeling applies homology to identify sequence similarity, the predicted conformation is more accurate if the homologous relationship is established. Rating the compatibility of the unknown protein to known structures, the fold recognition methods are more applicable if the sequence similarity can not be established.

Unlike the other two methods, the *ab initio* method does not use conformational information from the PDB but uses a force field model which has parameters empirically developed from the PDB. The *ab initio* method usually determines conformation using a molecular dynamic simulation or an optimization search. To improve the performance of dynamic simulations and optimization approaches, researchers have focused on improving both the molecular and energy models and the dynamics and optimization algorithms.

This literature review is organized as follows: Sections 2.1 and 2.2 discuss work on models, namely energy and reduced molecular models respectively. Sections 2.3 and 2.4 discuss work on conformation prediction methods, namely molecular dynamics and optimizations respectively. Then Section 2.5 discusses work on energy landscaping that improves optimization, and lastly Section 2.6 concludes the literature review.

2.1 Energy Models

In addition to the molecular models, a protein simulation requires an energy model that describes potential energies between atoms. These energies are important for evaluating a protein configuration in the protein prediction process. Moreover, atomic forces can be determined from the negative of energy gradients.

Two basic energy modeling approaches are physics based and knowledge based [65]. While the physics-based approaches are governed by the laws of physics, the knowledge-based approaches use potential energies with sequence-independent and sequence-specific terms that are empirically derived from the protein structures in the PDB library. In protein conformation prediction the knowledge-based approaches are more successful than the physics-based approaches [65] as they can obtain a configuration closer to the native state; however, the resulting configurations are not necessary the native conformation. Research on energy-landscape sculpting and an improved structure selection has produced encouraging results, but further improvements are still needed [65].

2.1.1 Force Field Model

Molecular chemists have modeled intra-molecular reactions with force field models, which govern the dynamic motion of the folding process in order to calculate the potential energy and internal forces of the protein. Expressing the potential energies of a protein molecule as functions of the molecular structure, the force field model reduces the computational costs over a quantum mechanics approach but is

less accurate [44]. In attempts to describe the quantum mechanics of a molecule, parameterized functions in the force field model are empirically found. Although the force field model can sometimes yield results that are as accurate as those obtained from quantum mechanics, it cannot produce accurate values for properties affected by electronic distribution in the molecule, such as electrostatic energy.

Typically force field models fall into two types, all-atom and coarse-grained models. The all-atom force field model describes potential energies that include bond, bending (bond angle), torsion, van der Waals, and electrostatic energies. Since at least one of these energies occurs between every pair of atoms, this model is used with a molecular model that explicitly describes the positions of most atoms. In contrast, when the molecular model has several atoms lumped into larger elements and when atom positions are not explicit, a coarse-grained force field model is used. This model describes the potential energies between the larger elements and these terms are different from the energy terms in the all-atom force field model. Moreover, parameters in individual functions describing each potential energy in one all-atom force field model are not transferable to another force field model because the parameterization of all potential energies in a model are dependent [60].

The most widely accepted all-atom force field models are those of Charmm [12] and Amber [16] which are frequently used as bases for other advanced models. Improving the energy calculation for large molecules, a modified force field [20] reduces the number of atoms by replacing aliphatic and aromatic CH3, CH2, and CH groups with several types of single, united atoms. Similarly, the new-generation Amber united-atom force field model [77] reduces the force calculations in dynamic simulations by assuming that the positions of hydrogen atoms bonded to an aliphatic carbon atom are the same as the position of the carbon atom. Further improvement can be obtained using the coarse-grained molecular model, which reduces the degrees of freedom of a protein more than the united-atom model. However, the coarse-grained force field

model for one type of element is not transferable to another. While all-atom force field models can be used with any molecular models in which explicit atom positions can be found, the coarse-grained force field models do not have the same flexibility.

2.2 Reduced Molecular Models

2.2.1 HP Model

To reduce the complexity of the protein-analysis problem, researchers generally use a reduced protein model. For example, sequence identification performed by searching in a sequence space is a hard combinatorial optimization problem because its sequence space has 20^N possible sequences for a protein with N amino acids. Therefore, most studies of this problem have been done with smaller search spaces using HP and lattice models. The HP model reduces the sequential space by classifying amino acids into only two groups, hydrophobic (H) and polar (P); a residue-level, self-avoiding, compact lattice model reduces the conformation space by allowing a residue to have at most three interactions with other residues for a 2D model and five interactions for a 3D model [43]. The force model also becomes much simpler. However, even with these models the combinatorial optimization search is difficult. Koh *et al.* [41, 42] avoided this difficulty by defining a continuous state energy function of the simplified protein so that a continuous optimization algorithm could be used. Besides sequence identification, HP and lattice models are widely used in a conformation-identification problem, such as protein folding [78]. However, they are not applicable to tertiary structure identification because of their coarse resolution.

2.2.2 Residue-Level Models

The residue-level model is based on an assumption that the peptide planes are rigid. Modeling a protein as a series of rigid bodies connected by revolute joints makes it possible to use the kinematic model of a serial manipulator that is used in robotics [36, 81]. For example, Kazerounian *et al.* explicitly described the kinematics of the

protein, including both a main chain and side chains [36], using the zero-reference position notation [25, 26] which defines joint angles from a user-defined zero position. This notation supposedly requires less computation than other notations. They have proposed improved intrinsic molecular parameters, such as bond angles and lengths, which are more suitable for the residue level rigid-chained protein model [69]. They have also developed an inverse kinematics algorithm to calculate dihedral angles [68]. Their algorithm is compared with other algorithms by deriving atom coordinates from the forward kinematics of their rigid-chain molecular model. However, since other methods do not use the same molecular model to derive dihedral angles, this comparison may be biased. Using one form of Denavit-Hartenberg parameter notation in robotics and modeling larger atom clusters as rigid bodies, Zhang and Kavraki [81] developed a method to derive a molecular conformation. Although this method is supposedly fast, its advantage over the others is not clear because the study did not directly compare the results using the same large clustered atom model.

Besides a reduction in the degrees of freedom, the residue-level model also reduces computational cost by simplifying the force field models [36]. Because of the rigidity assumption, bond lengths, bending angles, and torsion angles are assumed to be unchanged. The Amber force field terms are reduced to van der Waals and electrostatic energies because other energies are constant and they do not affect atomic forces. The mathematical formulation of the Amber force field [34] is

$$E_{ij} = \sum_{vdW} \left(\frac{A_{ij}}{R_{ij}^{12}} - \frac{B_{ij}}{R_{ij}^{6}} \right) + \sum_{elect} \left(\frac{q_i q_j}{\epsilon R_{ij}} \right) \tag{2.1}$$

where E_{ij} is the potential energy between atoms i and j, A and B are experimentally determined constants, R_{ij} is the distance between the two atoms, q_i and q_j are the charges on each atom, and ϵ is the dielectric constant.

2.2.3 Macro-Level Models and Rigidity Analysis

While residue-level models greatly reduce the degrees of freedom of a protein chain, macro-level models have been developed to further reduce the degrees of freedom of a known protein. This reduction is done by identifying the rigid parts of the protein chain using a rigidity or flexibility analysis. A simple method for determining rigid clusters is to compare the different conformations of the same protein in the PDB library. A rigid cluster is a region where the dihedral angles are the same in all conformations. Using this method and a rigid-body-based model, Kim *et al.* [39] generated a feasible path between two conformations. A more sophisticated rigidity analysis uses a graph theoretical technique. Analyzing a molecule modeled as a bar-joint network, Thorpe *et al.* [30, 71] used the "pebble game" to identify the over- or under-constrained regions of a network that correspond to rigid and flexible regions of the molecule. They have also developed the computer program FIRST (Floppy Inclusion and Rigid Substructure Topography) [29], which identifies rigid clusters and flexible substructures in macromolecules. After the rigid regions (called "ghost templates") are defined their new computational method FRODA (Framework Rigidity Optimized Dynamic Algorithm) [75] can be used to dynamically determine the protein motions. Although the rigidity analysis is a great tool to reduce the computational time in simulations it requires information from the PDB.

2.3 Molecular Dynamics (MD)

2.3.1 Traditional Molecular Dynamics

Two common *ab initio* techniques for the protein-folding problem are the molecular dynamics (MD) simulation and the optimization search. Although in theory MD simulations can be used to locate the native state, they are typically used to study detailed kinetics or dynamics of the protein folding mechanism [14]. The main limitation of MD simulations for protein folding is the required computational power [7, 14].

Since a typical MD simulation is done iteratively by computing the atomic forces on each atom, the acceleration, and then the displacement numerous calculations are required for each iteration, even for a small protein. Moreover, another computational problem is caused by the requirements of a small time step and a lengthy simulation. Since the force functions are highly non-linear the time step of MD simulation must be small, typically in fs for the constant-acceleration approximation between increments to be accurate [60]. Because the protein-folding process usually takes a relatively long time, typically in ms or s, the simulation of protein folding is nearly impossible. Because of these limitations, most MD simulations of the folding process are implemented in small fragments of protein molecules [14] and the MD simulations of full-size proteins are generally used in studies of protein motions. Because the MD simulation of the folding process of a full-size protein requires significant computational time, it can not be done in a reasonable amount of time with current computational power.

2.3.2 Internal Coordinates

By introducing internal coordinates, or configuration variables, many robotic dynamics algorithms can be applied in order to reduce the computational cost in molecular dynamics computations. As bond angles and lengths are fixed, the cost reduction is due to the decreasing number of the degrees of freedom of a protein chain. Also efficient forward dynamics algorithms [23, 24, 31, 53, 55] are already developed in robotics. For example, a traditional inertia-matrix algorithm (IMA) solves an equation of motion, $H\ddot{q} = \tau - C$, where H is a joint-space inertia matrix (JSIM), $\ddot{q}$ is a joint acceleration vector, τ is a joint force vector, and C is a bias vector containing velocity related terms. While an efficient method to calculate H, such as a composite-rigid-body algorithm (CRBA), is available the IMA still has a computational complexity of $O(N^3)$, of which N is the degrees of freedom of the protein molecule. A complexity

of only $O(N)$ can be achieved with an articulated-body algorithm (ABA) which describes an applied force as a linear function of an acceleration for an unconstrained or floating chain of bodies. As a result, the CRBA becomes less efficient than the ABA, when $N > 9$ [21, 24]. Another JSIM calculation in the IMA that is more efficient than CRBA is a sparse factorization algorithm [23]. This exploits the sparsity of the JSIM of a branched kinematic tree such as a protein structure and the fact that sparsity and symmetry are preserved. Although the reduced complexity is between $O(N)$ and $O(N^3)$, the IMA is still less efficient than the ABA because the ABA can be applied to a branched chain structure without increasing complexity. Another drawback of the IMA is that the JSIM becomes ill-conditioned as N increases [22].

Jain *et al.* [31] suggested a fast recursive IMA for molecular dynamics simulations called "Newton-Euler Inverse Mass Operator (NEIMO) dynamics." Assuming that the base of the molecule has six degrees of freedom, which is equivalent to floating, NEIMO can be proved equivalent to the ABA using spatial operators [24, 31], which has only $O(N)$ complexity. In [47], NEIMO is implemented in several polypeptides along with the cell multipole method (CMM). The CMM improves the computational time of non-bonded interactions. By dividing the molecule into small cells, the algorithm evaluates the non-bonded interactions only if the atoms are in the same or adjacent cells; otherwise the interactions are estimated. The CMM is more accurate than the cut-off method, in which the non-bonded interactions of distant atoms are ignored. Both NEIMO and CMM reduce the computational time and increase the simulation time step, but not significantly enough to solve the protein-folding problem in large proteins. Since proteins and their solvents in typical chemistry experiments are in contact with a constant-temperature heat bath, [72] presents constant-temperature constrained molecular dynamics simulated by solving NEIMO-Nosé and NEIMO-Hoover equations. In [8], NEIMO is used to study the α-helix formation,

which provides some useful insight in protein folding. With a rigid-link method identical to NEIMO, Rapaport [53] also folds some α-helices. However, in addition to a high-energy initial configuration, the simulated annealing technique [40] is used as the temperature of the simulated system is reduced by velocity scaling. The simulated annealing technique is explained in Section 2.4.2. Although this algorithm results in a higher success rate of 85% compared to NEIMO, the torsional potential derived from the native configuration of an α-helix may be a main contribution.

2.3.3 Normal Mode Analysis

Since the force field model requires a large number of computations, a normal mode analysis (NMA) has been widely used to approximate interactions between atoms or residues as the dynamics of a molecule can be represented by the sum of vibration modes. The NMA assumes that point masses are restrained to C_α atoms at which the side chains are located, and it models a protein molecule as a cluster of rigid bodies connected by springs but not dampers. Although the real motion of the protein is highly damped, the NMA has been widely used. Alexandrov *et al.* [2] have statistically analyzed the applicability of the NMA to protein-conformation prediction and found little correlation between a single normal-mode or lowest-frequency vibration and protein motion.

One type of the NMA is the elastic-network model in which all springs have equal stiffness and only bodies within a cutoff distance are considered connected. The work of Kim *et al.* [38] on C-alpha coarse-grained elastic network interpolation shows a feasible path between two conformations. Changes between intermediate conformations are chosen such that potential energy is minimized. An extension of this work, rigid-cluster coarse-grained elastic network interpolation (ENI) [39], uses a macro-level protein model whose clusters are formed by rigidly-attached point masses. Since the number of rigid bodies in the protein chain decreases and because

only six linear springs on each cluster are necessary to represent rigid-body motions, the number of interactions significantly decreases. The hybrid ENI has rigid clusters for the rigid parts of the chain and point masses on C_α atoms for fairly flexible parts. Although various NMA models may prove useful in the simplification of the atomic interactions they are not applicable to protein folding [2].

2.4 Optimizations

2.4.1 Classical Optimization Methods

Since the MD simulation cannot solve the protein-folding problem with currently available computation resources, an optimization algorithm that searches for a minimum energy conformation is frequently used [60]. However, the performance of an optimization algorithm greatly depends on the formulation details of the problem, such as potential functions, the protein model, and the type of a protein, so a performance comparison between algorithms is sometimes difficult. Although numerous algorithms were identified for the protein-folding problem, all are variations of an analytical or statistical optimization algorithm, or a combination of both.

Examples of analytical optimization algorithms are a gradient method called "the method of steepest descent (SD)," Newton's method, and Newton-based methods. While the SD is simple to implement and not computationally expensive its convergence is slow. Newton's method provides a quadratic convergence rate which is superior to the linear rate found in the SD method [18, 49]. However, the second derivative of the potential energy can be computationally expensive for the all-atom model and in many cases it may not be available [60]. Both SD and Newton's method are more robust when a one-dimensional optimization called a "line search" (described below) is implemented, but the computation time can significantly increase. A line search is not necessary in some methods such as the quasi-Newton (QN), the nonlinear conjugate gradient (CG), the truncated Newton (TN), and the traditional SD

methods because they are designed to provide a proper step size. QN methods such as the Davidon-Fletcher-Powell (DFP) and the Broyden-Fletcher-Goldfarb-Shanno (BFGS) methods, except for the limited-memory type, require extensive memory storage. This problem does not occur in the CG and TN methods; however, the CG method converges more slowly than the QN method and the implementation of the TN method is more complex. Compared with the other methods mentioned above Newton's method is the most reliable with the fastest convergence despite its expensive computation.

2.4.1.1 Line Search

After a descent direction in which the cost function locally decreases is determined by the SD, the Newton's, or the QN methods, a line search along this direction determines a scalar multiplier here called a "line search gain" that locally minimizes the cost function [6, 60]. An example of a very simple line search is a bracketing algorithm [6] that advances the search step with a varying or a constant interval in monotonically decreasing direction until the function no longer decreases. Unlike the bracketing algorithm that expands the search interval, interval reduction methods, such as the Fibonacci and the golden section methods, reduce the search interval until a stopping criterion is met. Requiring a minimum inside an initial search interval, these algorithms ensure that reduced intervals contain a minimum as well. The Fibonacci and the golden section methods reduce the interval such that a ratio between consecutive intervals equal to the ratio between two consecutive Fibonacci numbers or the golden ratio, respectively. A more robust algorithm is an integration of a bracketing algorithm and the golden section method. With a bracketing technique this integrated algorithm identifies an initial interval by using the golden ratio to determine an advancing step and the interval is reduced by the golden section method. Because of a slow convergence this algorithm requires a large number of

function evaluations and can be computationally expensive. To improve efficiency it is combined with a polynomial fit-sectioning algorithm. For example, Brent's method performs quadratic fitting on a bracketed interval and determines a minimum from the obtained quadratic equation when applicable and it uses golden section method otherwise. Less frequently used than a quadratic fit, a cubic and a higher polynomial fits require more points or derivative information so they are inefficient if not inapplicable when the derivative along the descent direction is unavailable.

2.4.2 Monte Carlo

One of the most popular statistical optimization algorithms used in the protein-folding problem is the Monte Carlo (MC) algorithm [44]. MC simulation generates statistical ensembles (or configurations) by adding random changes to the current position. These changes are between the maximum displacements $\pm\delta r_{max}$ and the acceptance of the new configuration is determined by the Metropolis criterion [48]. If the new configuration has a lower potential energy it will be accepted as a new current position. If it has a higher potential energy it will be rejected except when the Boltzmann factor is greater than or equal to a random number between 0 and 1. The Boltzmann factor is defined as $exp\,(-\Delta E/k_B T)$, where ΔE is the energy difference, k_B is the Boltzmann constant, and T is the temperature. This exception allows the higher-energy ensembles to be accepted at some probability. The smaller the value of ΔE, the higher the probability of acceptance. As the Boltzmann factor determines the probability of an ensemble, the δr_{max} controls the rate of change in the ensemble generation. When δr_{max} is small, the simulation will generate many acceptable ensembles but they may not be far away from the initial position. When δr_{max} is large, the simulation will generate very different ensembles but most of them may be rejected. Therefore, the difficulty of the MC algorithm is finding an efficient value of δr_{max} for each protein.

As most generated ensembles will be rejected, especially for a large protein, the MC algorithm can be inefficient. Therefore, a combination of MC and other techniques is generally used in practice. For example, a hybrid MC algorithm [60] uses MD to bias the ensemble generation and the MC criterion to accept or reject the new ensemble. Another example [44] is MC algorithm with simulated-annealing technique [40], inspired by manufacturing process that strengthens materials. The simulated-annealing process for conformation prediction starts at a high temperature and ends at absolute zero. While also applicable to MD and MC with other algorithms, this technique uses MC algorithm to find a thermal equilibrium configuration at each temperature. Once the temperature drops to the absolute zero, the molecule should be at the global minimum energy conformation. However, this conformation is not guaranteed because at each temperature step thermal equilibrium must be achieved and an infinite number of steps are required. Therefore, in practice several different runs are done to increase the probability of finding the global minimum energy conformation.

2.4.3 *De Novo* Methods

Compared to *ab initio* techniques for solving the protein-folding problem *de Novo*, or knowledge-based methods, that use information from the PDB have better performance. Here two representative examples are discussed.

2.4.3.1 TASSER

Zhang and Skolnick [84, 83, 85] reduce the degrees of freedom of a protein by structure alignment that utilizes the PDB to identify templates or secondary structures of a protein. A part of an unknown protein with a 35% sequence identity to a secondary structure of a known protein in the PDB yields a template which assumes the shape of that structure. Then several of these templates construct an initial template alignment that was used as an initial structure of a conformation search done by MC sampling. The internal configurations of the templates are kept constant while the

configuration of the residues in the gapped region (between templates) is generated using the *ab initio* lattice modeling approach. A representative benchmark study [85] on 1,489 single-domain, medium-size proteins in the PDB with 41 to 200 amino acids shows that a template with the root-mean-square deviation (RMSD) from the native state of 2.5 Å can always be found. While the average RMSD value of the full-length models is 2.25 Å, 97% of them have RMSD values below 4 Å. Another benchmark study [84] is on 745 medium-to-large-size proteins with 201 to 300 residues. The results show that the templates of 365 cases have an RMSD value of 6.5 Å or less with 70% alignment coverage. Furthermore, 408 of the full-length models have the RMSD values lower than 6.5 Å.

2.4.3.2 Rosetta

Rosetta [11, 56] predicts the protein conformation using a *de Novo* method that constructs several configurations called "decoys" from pre-generated fragments and then a refinement protocol optimizes the predicted conformation. As an energy function, Rosetta uses a potential energy surface (PES) called "scoring function" which is derived based on a Bayesian separation of the total energy [63, 64]. Using the Monte Carlo-minimization approach [46], also called "basin-hopping" [73], the refinement protocol applies a series of random changes and minimization to each decoy. More details about this software and its method are discussed in Chapter 5. Bradley *et al.* [11] achieves high-resolution structure prediction for sixteen small proteins which have less than 85 residues. Five proteins have C_α-RMSD less than 1.5 Å and eight proteins have 1.5 to 5.0 Å but no native side-chain packing. Since 20,000 to 30,000 generated decoys are necessary for each protein, this process is very computationally expensive, as it takes about 150 CPU days per molecule. The improvement in the efficiency of the refinement protocol can significantly impact the performance of the method and reduce the computational time.

2.4.4 Non-Traditional Optimization

Besides the family of Newton's methods and MC algorithms, many studies have used non-traditional approaches. For example, Yoon [78] applied integer programming and constraint programming models to protein folding using HP and lattice models. Although his models are not fast enough to be used in practice, they showed superiority over comparison algorithms. Another exciting approach is a control approach even though it has not been widely explored. Most control approaches applied to protein folding are generic or evolutionary algorithms, as in [19], but a study of protein folding using other control techniques, such as classical and adaptive controls, is rare. Canutescu and Dunbrack [15] use an interesting control approach in "protein loop closure" which determines if a structure can geometrically close a protein loop. Fixing one end of the structure to one end of the protein and changing the dihedral angles of the structure one at a time, their cyclic coordinate descent (CCD) algorithm verified if another end of the structure could possibly meet another end of the protein. Since CCD algorithms check only the geometric feasibility of the structure but not the energetic compatibility between the combined compounds, it is not applicable to the protein-folding problem.

2.5 Energy Landscape Modification

The complexity of the energy function results in numerous local minima, at which the optimization schemes get trapped. Addressing this challenge, some studies work on the modification of the energy landscape so that it assists the minimization algorithm.

2.5.1 Hypersurface Deformation

An approach of energy landscape modification, called "hypersurface deformation", has been investigated for at least three decades [4, 5, 13, 50, 51, 59, 66, 74]. In general, these algorithms smoothen the landscape by reducing the local minima, which aids

the global minimum search. However, the obtained global minimum has to be mapped back to the original surface, which can be troublesome and unreliable [73].

In a global optimization scheme, Azmi *et al.* [4, 5] use a non-integration based spatial averaging to smooth the non-bonded interactions of the potential energy function removing local minima while maintaining the basic structure of the energy landscape. However, the smoothing method alters the locations of the minima, so several desmoothing steps that perform a local minimization on the original energy landscape are required. Although this optimization scheme successfully predicts the conformation of a 70-residue helical protein, the smoothing method is not applicable to other energy functions but only to Lennard-Jones and electrostatic energy functions.

Based on an idea that the energy landscape is a funnel with bumps, the Convex Global Underestimator (CGU) global optimization algorithm [50] iteratively forms a least-underestimated separable-quadratic surface under minima found by sampling the energy landscape. After finding the global minimum on the obtained surface, it locates the corresponding local minimum on the energy landscape. Then more samplings are focused to an area near the minimum of the surface and the surface is regenerated. The global minimum is found if it is equal to its corresponding local minimum. The algorithm appears to work well on simple homopolymers and simplified Sun energy functions that are funnel-like. However, the method of determining a successful run is questionable. Since the global minima for most proteins are unknown, a run is considered successful if the predicted global minimum is within 1% of the lowest energy value sampled from the energy landscape. Moreover, the CGU algorithm becomes much less effective on real proteins and more complex energy functions because the underestimated surface does not represent the energy landscape very well [50].

Since most methods, such as in [50], require intensive samplings in order to find the global solution, Burke and Yaliraki [13] use sum of squares (SOS) decomposition

to transform a nonconvex function in Euclidean space to a nonnegative convex SOS function in the vector space of monomials. Without sampling the global minimum, or a solution close to the global minimum, can be found from the SOS function which is a much easier minimization problem. Global minimization on several potentials with similar shapes to a potential energy surface is presented. However, since these potentials are represented only as polynomial equations, this work may not be applicable to the actual energy functions unless their polynomial forms can be approximated.

2.5.2 Energy Landscape Flattening

Encouraging the advancing of the Monte Carlo optimization step, parallel hyperbolic sampling (PHS) flattens the energy landscape by shortening high energy barriers with little effect on lower energy barriers [87]. With the same computation time, the PHS results in lower energy configurations than the replica sampling algorithm [27, 70], which changes the state of the low-energy configuration so that the Monte Carlo step can pass over high-energy barriers. This work, however, does not study a comparison between the flattened and unmodified energy landscape. Also, since the PHS can be used with only the Monte Carlo method, other methods of energy landscape modification are required for other optimization algorithms.

2.6 Conclusion of the Literature Review

Due to the inapplicability of the molecular dynamic simulation, several optimization algorithms including both probabilistic and analytical methods are used in the protein folding problem instead. While analytical methods can locate local solutions very well, by themselves they are not successful in a very complex and nonlinear optimization problem. Therefore, the ability to search a global space with probabilistic methods is a very useful addition to the analytical methods. Hence, the most successful algorithms, *de Novo* algorithms, perform exceptionally well because they use the knowledge obtained from the protein database and combine both probabilistic

and analytical methods.

Although the *de Novo* methods can be enhanced by improving the efficiency of either the probabilistic methods or the analytical methods, improvements in the latter methods are more significant because of two following reasons: 1) they are the bottleneck of the *de Novo* methods as they take a longer time, 2) they involve in the majority of the process as they are intensively used in refinement protocols which repeatedly perform local minimization. Reducing the computational time required for the analytical algorithms will significantly improve the overall processing time of the *de Novo* methods.

The literature review shows that the energy landscape modification is a promising method that can be used to improve the efficiency of the analytical algorithm. Yet the method has never been thoroughly investigated. The hypersurface deformation method suggests that the energy landscape modification is beneficial but altering the minima's locations requires other troublesome tasks. Without the alteration, the energy landscape flattening provides an improvement to the Monte Carlo method but not analytical algorithms. In most research on the energy landscape modification a comprehensive study is needed to fully understand its effect.

In conclusion, the objective of this research is to develop a method of energy landscaping that enhances the efficiency of the analytical algorithms for the protein prediction problem. Since Newton's method provides the basis for several powerful analytical methods that are commonly used in conformation prediction, it is chosen to be investigated along with two quasi-Newton methods. A comparative study of the effect of an energy landscaping method on several variations of these algorithms is performed.

CHAPTER III

NEWTON'S METHOD AND QUASI-NEWTON METHODS APPLIED TO PROTEIN FOLDING PROBLEM

This chapter discusses basic concepts of Newtonian optimization and its application to the protein folding problem. Sections 3.1 and 3.2 discuss the derivation of Newton's method and two quasi-Newton methods for an optimization problem. Section 3.3 discusses how these methods are applied to the protein folding problem. Lastly, Section 3.4 summarizes and concludes the chapter.

3.1 Newton's Method

Newton's method is often used to locate the root of a function, but for minimization problems it is used to locate the roots of the derivatives of the cost function f giving a stationary value of the cost function. Newton's method can be derived from the Taylor series expansion of the cost function derivative,

$$\frac{\partial f(\theta_k + h)}{\partial \theta} = \frac{\partial f(\theta_k)}{\partial \theta} + \frac{\partial^2 f(\theta_k)}{\partial \theta^2} h + \frac{1}{2} h^T \frac{\partial^3 f(\theta_k)}{\partial \theta^3} h + \dots$$

where $h \equiv \Delta\theta$ is a small change in the vector variable θ. Dropping the third- and higher-order terms and setting $\dfrac{\partial f(\theta_k + h)}{\partial \theta} = 0$ gives

$$0 = \frac{\partial f(\theta_k)}{\partial \theta} + \frac{\partial^2 f(\theta_k)}{\partial \theta^2} \hat{h}$$

where $\hat{h}$ is an approximation of h. Rearranging gives

$$\hat{h}_k = -K_k^{-1} \frac{\partial f(\theta_k)}{\partial \theta} \tag{3.1}$$

where $\hat{h}_k$ is the k^{th} descent direction of the Newton's method, here called "Newton step," and $K_k \equiv \dfrac{\partial^2 f(\theta_k)}{\partial \theta^2}$ is the Hessian matrix. The $(k+1)^{\text{th}}$ approximated solution becomes

$$\theta_{k+1} = \theta_k + \hat{h}_k$$

Newton's method fundamentally gives a solution to the quadratic approximation of a function [18]. Therefore, it can achieve a quadratic convergence rate superior to the linear rate found in the gradient methods, such as the steepest descent method [18, 49]. Generally, the Newton's method can locate a solution only if it is close to the starting point. Also if the Hessian matrix is not positive definite or if it is singular the Newton step may worsen the optimization or the Newton step may not exist [6]. To avoid divergence, Newton's method is commonly implemented with a line search [6, 49].

3.1.1 Line Search

Usually implemented with Newton-based algorithms, a line search determines a line search gain or a scalar multiplier of the step.

$$\alpha_k = \texttt{line\_search}(\hat{h}_k)$$

where α_k is the gain that minimizes $f(\theta_k + \alpha\hat{h}_k)$. The revised $(k+1)^{\text{th}}$ approximated solution becomes

$$\theta_{k+1} = \theta_k + \alpha_k \hat{h}_k \tag{3.2}$$

where $\alpha_k \hat{h}_k$ is the total step.

It is impractical to globally minimize $f(\theta_k + \alpha\hat{h}_k)$ so the gain only gives a solution on a limited interval. Many robust line search algorithms define an initial value of α equal to 1 so the magnitude of the Newton step $|\hat{h}_k|$ sets the size of the initial search interval which means different values of $|\hat{h}_k|$ yield different searched locations. Since the gain only locally minimizes f in the direction of the Newton step, different

Pseudo-code: Newton's method with a line search

Initialize ϵ, θ_0, and θ_1

Calculate $f(\theta_0)$

for $k = 1, \ldots, k_{max}$ **do**

Calculate $f(\theta_k)$, $\dfrac{\partial f(\theta_k)}{\partial \theta}$, and $\dfrac{\partial^2 f(\theta_k)}{\partial \theta^2}$

if $\left| \dfrac{f(\theta_k) - f(\theta_{k-1})}{f(\theta_{k-1})} \right| < \epsilon$ **then**

`break` // Convergence criterion met.

end if

$\hat{h}_k = -\left(\dfrac{\partial^2 f(\theta_k)}{\partial \theta^2} \right)^{-1} \dfrac{\partial f(\theta_k)}{\partial \theta}$ // Determine the Newton step

$\alpha_k = \texttt{line\_search}(\hat{h}_k)$ // Determine the line search gain

$\theta_{k+1} = \theta_k + \alpha_k \hat{h}_k$ // $(k+1)^{\text{th}}$ solution

end for

Figure 3.1: A pseudo-code for Newton's method with a line search

searched locations can result in different gains as they correspond to different local minima. Figure 3.1 shows a pseudo-code for the Newton's method with a line search.

3.2 Quasi-Newton (QN) Methods

Since the calculation of the Hessian matrix K can be computationally expensive or is often unavailable, quasi-Newton (QN) methods use only the first derivative to estimate K by satisfying the secant equation,

$$\frac{\partial f(\theta_{k+1})}{\partial \theta} - \frac{\partial f(\theta_k)}{\partial \theta} = \hat{K}\hat{h} \tag{3.3}$$

where $\hat{K}$ is the estimated Hessian matrix. Equation (3.3) is underdetermined so many methods apply other criteria to find the solution. Minimizing the Frobenius

norm $\|K_{k+1} - K_k\|_{\mathrm{F}}$, Broyden's method [17] recursively estimates K,

$$\hat{K}_k = \hat{K}_{k-1} - \left(\hat{h}_{k-1}^T \hat{h}_{k-1}\right)^{-1} \left(\gamma_{k-1} + \hat{K}_{k-1}\hat{h}_{k-1}\right) \hat{h}_{k-1}^T \tag{3.4}$$

where

$$\gamma_{k-1} \equiv \frac{\partial f(\theta_k)}{\partial \theta} - \frac{\partial f(\theta_{k-1})}{\partial \theta}$$

is the change in the gradient of the cost function f. The QN step becomes

$$\hat{h}_k = -\hat{K}_k^{-1} \frac{\partial f(\theta_k)}{\partial \theta}$$

$$\theta_{k+1} = \theta_k + \alpha_k \hat{h}_k$$

Figure 3.2 displays a pseudo-code for a QN method with a line search that uses Broyden's method to estimate K.

While Broyden's method works fairly well, it does not guarantee a symmetric and positive definite $\hat{K}$ [17]. Since the Hessian matrix is alway symmetric and often positive definite, the Broyden-Fletcher-Goldfarb-Shanno (BFGS) algorithm ensures these properties as it estimates K^{-1} by satisfying the quasi-Newton condition [6]

$$\hat{H}_k \gamma_k = \hat{h}$$

where $\hat{H}$ is an estimation of K^{-1}. Widely used in an nonlinear optimization, the BFGS algorithm is one of the most powerful analytical optimization algorithms when used with a line search. The BFGS algorithm recursively calculates $\hat{H}$ as,

$$\hat{H}_k = \hat{H}_{k-1} - \left(\frac{\hat{H}_{k-1}\gamma_{k-1}\hat{h}_{k-1}^T + \hat{h}_{k-1}\gamma_{k-1}^T \hat{H}_{k-1}}{\hat{h}_{k-1}^T \gamma_{k-1}}\right) + \left(1 + \frac{\gamma_{k-1}^T \hat{H}_{k-1}\gamma_{k-1}}{\hat{h}_{k-1}^T \gamma_{k-1}}\right) \frac{\hat{h}_{k-1}\hat{h}_{k-1}^T}{\hat{h}_{k-1}^T \gamma_{k-1}}$$

The k^{th} BFGS step $\hat{h}_k$ is defined as

$$\hat{h}_k = -\hat{H}_k \frac{\partial f(\theta_k)}{\partial \theta}$$

$$\theta_{k+1} = \theta_k + \alpha_k \hat{h}_k$$

Initializing $\hat{H}_0$ to be positive definite, BFGS algorithm guarantees positive definite $\hat{H}$'s. However, its direction can not guarantee to lower a function value because the

actual Hessian matrix may not be positive definite. Another drawback is that the QN methods may be slow to converge. The Hessian matrix is usually not constant and if the magnitude of the QN step $\hat{h}_k$ is too large $\hat{K}$ may fail to converge. Because of these drawbacks QN methods are usually implemented with a line search for best results.

Figure 3.3 gives a pseudo-code for the QN method and uses the BFGS algorithm

---

Pseudo-code: QN method with a line search and Broyden's method

---

Initialize ϵ, θ_0, θ_1, and $\hat{K}_0$

Calculate $f(\theta_0)$ and $\dfrac{\partial f(\theta_0)}{\partial \theta}$

for $k = 1, \ldots, k_{max}$ **do**

Calculate $f(\theta_k)$ and $\dfrac{\partial f(\theta_k)}{\partial \theta}$

if $\left| \dfrac{f(\theta_k) - f(\theta_{k-1})}{f(\theta_{k-1})} \right| < \epsilon$ **then**

`break` // Convergence criterion met.

end if

Calculate $\hat{K}_k$ // Broyden's method

$$\hat{h}_{k-1} = \theta_k - \theta_{k-1}$$
$$\gamma_{k-1} = \frac{\partial f(\theta_k)}{\partial \theta} - \frac{\partial f(\theta_{k-1})}{\partial \theta}$$
$$\hat{K}_k = \hat{K}_{k-1} - \left(\hat{h}_{k-1}^T \hat{h}_{k-1} \right)^{-1} \left(\gamma_{k-1} + \hat{K}_{k-1} \hat{h}_{k-1} \right) \hat{h}_{k-1}^T$$

$\hat{h}_k = -\hat{K}_k^{-1} \dfrac{\partial f(\theta_k)}{\partial \theta}$ // Determine the QN step

$\alpha_k =$ `line_search`$(\hat{h}_k)$ // Determine the line search gain

$\theta_{k+1} = \theta_k + \alpha_k \hat{h}_k$ // $(k+1)^{\text{th}}$ solution

end for

Figure 3.2: A pseudo-code for the QN method with a line search and Broyden's method

Pseudo-code: BFGS algorithm with a line search

Initialize ϵ, n, θ_0, θ_1, and $\hat{H}_0$

Calculate $f(\theta_0)$ and $\dfrac{\partial f(\theta_0)}{\partial \theta}$

for $k = 1, \ldots, k_{max}$ **do**

Calculate $f(\theta_k)$, and $\dfrac{\partial f(\theta_k)}{\partial \theta}$

if $\left| \dfrac{f(\theta_k) - f(\theta_{k-1})}{f(\theta_{k-1})} \right| < \epsilon$ **then**

`break` // Convergence criterion met.

end if

Calculate $\hat{H}_k$

$$\hat{h}_{k-1} = \theta_k - \theta_{k-1}$$

$$\gamma_{k-1} = \frac{\partial f(\theta_k)}{\partial \theta} - \frac{\partial f(\theta_{k-1})}{\partial \theta}$$

$$\hat{H}_k = \hat{H}_{k-1} - \left(\frac{\hat{H}_{k-1}\gamma_{k-1}\hat{h}_{k-1}^T + \hat{h}_{k-1}\gamma_{k-1}^T\hat{H}_{k-1}}{\hat{h}_{k-1}^T\gamma_{k-1}} \right) + \left(1 + \frac{\gamma_{k-1}^T\hat{H}_{k-1}\gamma_{k-1}}{\hat{h}_{k-1}^T\gamma_{k-1}} \right) \frac{\hat{h}_{k-1}\hat{h}_{k-1}^T}{\hat{h}_{k-1}^T\gamma_{k-1}}$$

$\hat{h}_k = -\hat{H}_k \dfrac{\partial f(\theta_k)}{\partial \theta}$ // Determine the BFGS step

$\alpha_k =$ `line_search`$(\hat{h}_k)$ // Determine the line search gain

$\theta_{k+1} = \theta_k + \alpha_k \hat{h}_k$ // $(k+1)^{\text{th}}$ solution

end for

Figure 3.3: A pseudo-code for BFGS algorithm with a line search

to estimate K^{-1} with a line search.

3.3 Application to the Protein Folding Problem

When optimization algorithms are applied to the protein folding problem, the cost function is the molecular energy function or force field, sometimes called the "score"

or "score function."

3.3.1 Energy and Derivatives Evaluation

In addition to van der Waals and electrostatic energies described in Equation (2.1), a force field for protein conformation prediction could include parameterized potentials that encourage a formation of secondary structures. These energy terms are of functions of different variables from molecular geometry such as the distance between atoms and molecular characteristics such as the amino acid type. Optimization with respect to the molecular geometry variables is a difficult constrained minimization problem so it usually done with respect to the dihedral angles and/or the Rotamer angles also called internal coordinates which is a non-constrained minimization problem. A closed form of the energy function in terms of these angles is however unavailable so transforming the angle information into molecular geometry information is needed before the function can be evaluated.

The expression of the first and second derivatives of the energy function with respect to internal coordinates are difficult to derive. Because energy functions are of functions of the distance between atoms, the chain rule is used. The first derivative calculation requires the derivatives of the distance between each atom pairs with respect to internal coordinates $\partial R_{ij}/\partial\theta_k$, where R_{ij} is the distance between i^{th} and j^{th} atoms and θ_k is the k^{th} angle in internal coordinates. These terms are difficult and inefficient to calculate because of two following reasons [45]: the number of atom pairs in a molecule are almost the square of the number of atoms and $\partial R_{ij}/\partial\theta_k$ is not zero if angle k is on the part of the chain joining i^{th} and j^{th} atoms. Also, since an average protein chain has several hundreds degrees of freedom, the calculation of the transformation becomes very inefficient. In practice only the first derivative is evaluated with various methods available [60] and the second derivative is estimated.

For energies described by molecular geometry such as the van der Waals and electrostatic energies, the first derivative of the energy (2.1) with respect to the distance R_{ij} between two atoms gives the attractive force F_{ij} acting on each atom [44]. The derivative of the energy with respect to each dihedral angle or Rotamer angle is the resulting torque with the same direction as the rotation vector of the angles contributed by these atomic forces. This is the sum of the dot product of the force and the resulting vector of a cross product between the rotational unit vector of the angle and the vector from the rotational vector to the force vector. The expression of the derivative of other energy terms are also available [44]. In this work the energy functions are from known simulated energy surfaces with known derivatives and from the Rosetta software package that also provides the first derivatives but not the second derivative.

The minimization variables of the energy function can be the dihedral angles, the Rotamer angles, or both. For coarse resolution minimization, which is usually done at the beginning of the prediction, the Rotamer angles are fixed at their most stable values and only the dihedral angles are subject to minimization. For full atom refinement, the Rotamer angles and/or the dihedral angles are subject to minimization. From this point forward this distinction will not be made as the variables of the energy function will be the dihedral angles only.

3.3.2 Newton's Method

Applying Newton's method to the molecular potential-energy function $f(\theta) = E$ gives the dihedral angle update vector $\hat{h}$ for energy minimization of a protein molecule. The first and the second order derivatives of the cost function are

$$\frac{\partial f(\theta)}{\partial \theta} \equiv \frac{\partial E(\theta)}{\partial \theta} = -\tau$$
$$\frac{\partial^2 f(\theta)}{\partial \theta^2} \equiv \frac{\partial^2 E(\theta)}{\partial \theta^2} = -\frac{\partial \tau(\theta)}{\partial \theta} = K$$

where θ is the dihedral angle vector, E is the potential energy, the negative gradient of E is the torque vector τ (τ_i is positive in the same direction as θ_i), and the Hessian of E is the stiffness matrix K. The Newton step (3.1) becomes

$$\hat{h}_k = K_k^{-1}\tau_k \tag{3.5}$$

For an analogous one dimensional rotational mass-spring system, this stiffness matrix becomes the stiffness K of a torsion spring at a joint connecting a body to the base, Figure 3.4. Away from the equilibrium state $\theta = 0$, the potential energy E stored in the spring is $E = \frac{1}{2}K\theta^2$ and the torque on the body is $\tau = -K\theta$. If the body rotates in the direction of the torque, opposite to the direction of the displacement angle, the potential energy in the spring decreases. This is analogous to Newton step (3.5) where the joint torques are scaled by the inverse of the stiffness matrix.

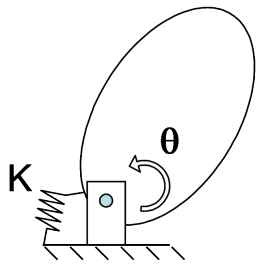

Figure 3.4: A simple rotational mass-spring system.

Some molecular energy functions contain non-energy terms that are empirically derived from statistical data in the protein database or from experimental data. Although the first and the second derivatives of these potentials are not torques or stiffnesses and may not have physical meanings, they are torque-like vectors and stiffness-like matrices, respectively. Therefore, the Newton update can still be applied to the score functions by expanding the definition of the derivative terms, τ and K, to include torque- and stiffness-like quantities.

3.3.3 QN Methods

Applied to the energy function E the QN step becomes

$$\hat{h}_k = \hat{K}_k^{-1}\tau_k \tag{3.6}$$

and γ is changed to

$$\gamma_{k-1} = \frac{\partial E(\theta_k)}{\partial \theta} - \frac{\partial E(\theta_{k-1})}{\partial \theta} = -\Delta\tau_{k-1}$$

where $\Delta\tau_{k-1} \equiv \tau_k - \tau_{k-1}$ is the change in the torque vector. so Broyden's method is rewritten as

$$\hat{K}_k = \hat{K}_{k-1} + \left(\hat{h}_{k-1}^T\hat{h}_{k-1}\right)^{-1}\left(\Delta\tau_{k-1} - \hat{K}_{k-1}\hat{h}_{k-1}\right)\hat{h}_{k-1}^T \tag{3.7}$$

From this point forward the "quasi-Newton algorithm (QNA)" implies QN step for the potential-energy function (3.6) that uses Broyden's method (3.7) to estimate the stiffness matrix.

Applying the BFGS to the energy function E yields

$$\begin{aligned}\hat{H}_k = \hat{H}_{k-1} &- \left(\frac{\hat{H}_{k-1}\Delta\tau_{k-1}\hat{h}_{k-1}^T + \hat{h}_{k-1}\Delta\tau_{k-1}^T\hat{H}_{k-1}}{\hat{h}_{k-1}^T\Delta\tau_{k-1}}\right) \\ &+ \left(\frac{\Delta\tau_{k-1}^T\hat{H}_{k-1}\Delta\tau_{k-1}}{\hat{h}_{k-1}^T\Delta\tau_{k-1}} - 1\right)\frac{\hat{h}_{k-1}\hat{h}_{k-1}^T}{\hat{h}_{k-1}^T\Delta\tau_{k-1}}\end{aligned} \tag{3.8}$$

The BFGS step $\hat{h}_k$ is defined as

$$\hat{h}_k = \hat{H}_k\tau_k \tag{3.9}$$

Equation (3.8) updates $\hat{H}$ using the current information of the energy function and the initial $\hat{H}_0$ is usually defined as an identity matrix.

3.4 Conclusion for Newton's Method and quasi-Newton Methods Applied to Protein Folding Problem

The derivation and the application to the protein folding problem of Newton's method, the QNA, and the BFGS algorithm have been discussed. In protein folding

Pseudo-code: Newton's method with a line search applied to the protein folding problem

---

Initialize ϵ, θ_0, and θ_1

Calculate E_0

for $k = 1, \ldots,$ k_{max} **do**

  Calculate E_k, τ_k, and K_k

  if $\left| \dfrac{E_k - E_{k-1}}{E_{k-1}} \right| < \epsilon$ **then**

    `break` // Convergence criterion met.

  end if

  $\hat{h}_k = K_k^{-1} \tau_k$ // Determine the Newton step

  $\alpha_k =$ `line_search`$(\hat{h}_k)$ // Determine the line search gain

  $\theta_{k+1} = \theta_k + \alpha_k \hat{h}_k$ // $(k+1)^{\text{th}}$ solution

end for

Figure 3.5: A pseudo-code for Newton's method with a line search applied to the protein folding problem

the estimation of the Hessian matrix $\hat{K}$ or its inverse $\hat{H}$ is most likely inaccurate and even the matrix K itself is not always positive definite due to the complex energy landscape. Even though a line search is used to ensure that the optimization moves toward a lower energy, the convergence can still be slow. For faster convergence a method of energy landscaping is presented in Chapter 4. Figures 3.5 thru 3.7 give pseudo-codes summarizing a Newton's method, the QNA, and the BFGS algorithm with a line search applied to the protein folding problem.

Pseudo-code: QNA with a line search applied to the protein folding problem

---

Initialize ϵ, θ_0, θ_1, and $\hat{K}_0$

Calculate E_0 and τ_0

for $k = 1, \ldots, k_{max}$ **do**

Calculate E_k and τ_k

if $\left| \dfrac{E_k - E_{k-1}}{E_{k-1}} \right| < \epsilon$ **then**

`break` // Convergence criterion met.

end if

Calculate $\hat{K}_k$

$$\hat{h}_{k-1} = \theta_k - \theta_{k-1}$$
$$\Delta\tau_{k-1} = \tau_k - \tau_{k-1}$$
$$\hat{K}_k = \hat{K}_{k-1} + \left(\hat{h}_{k-1}^T \hat{h}_{k-1}\right)^{-1} \left(\Delta\tau_{k-1} - \hat{K}_{k-1}\hat{h}_{k-1}\right) \hat{h}_{k-1}^T$$

$\hat{h}_k = \hat{K}_k^{-1}\tau_k$ // Determine the QNA step

$\alpha_k =$ `line_search`$(\hat{h}_k)$ // Determine the line search gain

$\theta_{k+1} = \theta_k + \alpha_k \hat{h}_k$ // $(k+1)^{\text{th}}$ solution

end for

Figure 3.6: A pseudo-code for QNA with a line search applied to the protein folding problem

Pseudo-code: BFGS algorithm with a line search applied to the protein folding problem

---

Initialize ϵ, n, θ_0, θ_1, and $\hat{H}_0$

Calculate E_0 and τ_0

for $k = 1, \ldots, k_{max}$ **do**

Calculate E_k, and τ_k

if $\left| \dfrac{E_k - E_{k-1}}{E_{k-1}} \right| < \epsilon$ **then**

`break` // Convergence criterion met.

end if

Calculate $\hat{H}_k$

$$\hat{h}_{k-1} = \theta_k - \theta_{k-1}$$
$$\Delta\tau_{k-1} = \tau_k - \tau_{k-1}$$
$$\hat{H}_k = \hat{H}_{k-1} - \left(\frac{\hat{H}_{k-1}\Delta\tau_{k-1}\hat{h}_{k-1}^T + \hat{h}_{k-1}\Delta\tau_{k-1}^T\hat{H}_{k-1}}{\hat{h}_{k-1}^T\Delta\tau_{k-1}} \right) + \left(\frac{\Delta\tau_{k-1}^T\hat{H}_{k-1}\Delta\tau_{k-1}}{\hat{h}_{k-1}^T\Delta\tau_{k-1}} - 1 \right) \frac{\hat{h}_{k-1}\hat{h}_{k-1}^T}{\hat{h}_{k-1}^T\Delta\tau_{k-1}}$$

$\hat{h}_k = \hat{H}_k\tau_k$ // Determine the BFGS step

$\alpha_k = \texttt{line\_search}(\hat{h}_k)$ // Determine the line search gain

$\theta_{k+1} = \theta_k + \alpha_k\hat{h}_k$ // $(k+1)^{\text{th}}$ solution

end for

Figure 3.7: A pseudo-code for BFGS algorithm with a line search applied to the protein folding problem

CHAPTER IV

ENERGY LANDSCAPING

This chapter presents a method of energy landscaping called the "exponential energy landscaping (XEL)" method. Unlike the hypersurface deformation discussed in the literature review this method only changes the heights and the depths of the extrema but not their locations. Section 4.1 presents the XEL method and its application to the Newton, the QNA, and the BFGS methods. Section 4.2 details characteristics of XEL algorithms and presents application guidelines. Section 4.3 derives two types of optimization algorithms for an adaptively modified energy landscape (AXEL). Lastly, Section 4.4 summarizes and concludes the chapter.

4.1 Exponential Energy Landscaping (XEL)

4.1.1 Method

With XEL the goal is to transform the energy landscape to facilitate the optimization search without changing the location of its minima and maxima. While these extrema in XEL are at the same locations they have different values. This avoids the troublesome remapping steps seen in hypersurface deformation algorithms. The modified energy function E^* is described by a nonlinear n^{th}-ordered exponential of the energy function E,

$$E^* = sign\,(E)\,|E|^n \tag{4.1}$$

where n, the exponential order of the energy function, is a positive real number, $n > 0$. The $|E|^n$ term modifies the magnitude of the energy value and the $sign\,(E)$ keeps the sign of the modified function the same as the energy function. This definition of the modified energy function is superior to E^n for two reasons. First, when n is

noninteger E^* is well defined for all energy values, unlike E^n. For example $E^{0.5}$ is undefined for negative E's. Secondly, E^* always preserves the locations of the extrema because it is always an odd function that preserves the sign of E. Unlike E^*, when n is an even integer E^n becomes an even function and a negative E gives a positive E^n value. This means negative minima and maxima become positive maxima and minima, respectively, and minima appear where $sign\,(E)$ changes.

Figure 4.1 illustrates XEL with the different values of n. When $n < 1$ the minima and the maxima are less defined as the energy landscape is flattened. This results in a smoother energy landscape and a larger step, which is shown in Section 4.2.4. A larger step can speed optimization at the beginning when the starting location is far from a minimum. The optimization path is also less likely to get stuck at shallow minima. On the contrary, when $n > 1$ the minima and the maxima are more defined

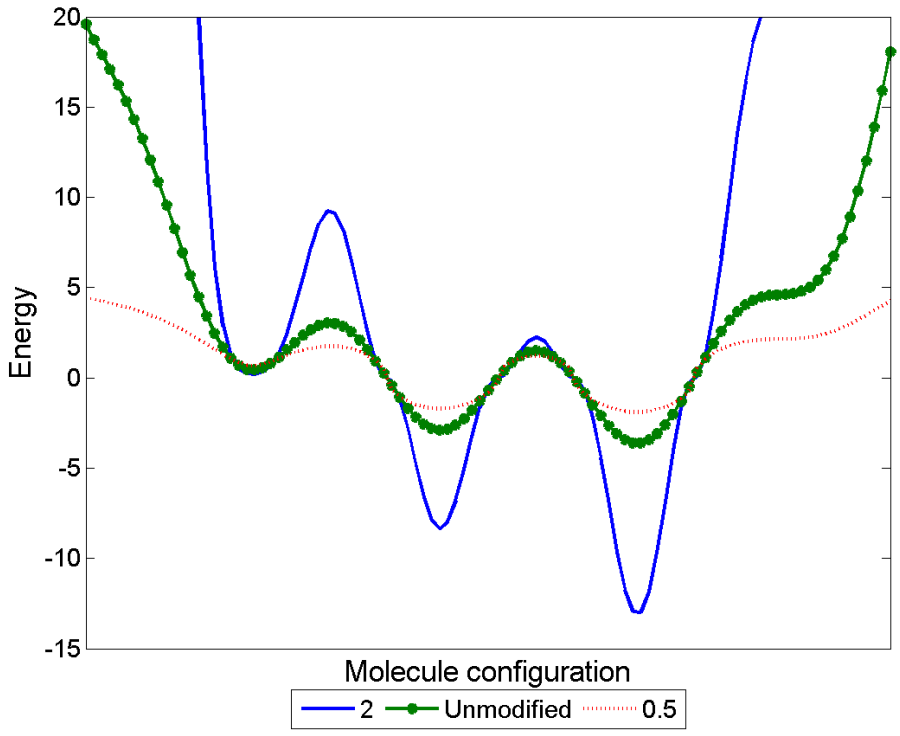

Figure 4.1: Comparison of unmodified energy landscape with $n = 1$, XEL with $n = 2$, and XEL with $n = 0.5$. The $n = 2$ XEL accentuates the extrema while the $n = 0.5$ XEL flattens the landscape.

as energy landscape is sharpened. This results in the more prominent minimum and the smaller step. This may be preferable toward the end of the optimization to ensure that the deeper minimum can be found.

Note that in the small range of $|E| < 1$ the effects of n on extrema are opposite to what described above. Figure 4.2 illustrates these effects. For $n < 1$, $|E|^n > |E|$ which means the energy landscape is sharpened. On the contrary, for $n > 1$, $|E|^n < |E|$ which means the energy landscape is flattened. Fortunately, because the value of the energy can be changed up to only ± 1 these opposite effects are not pronounced as they are unnoticeable in Figure 4.1. Also since most energy values on the energy landscape are less than 0, these opposite effects are less likely to happen during the optimization process.

The next three sections discuss the derivation of optimization algorithms with

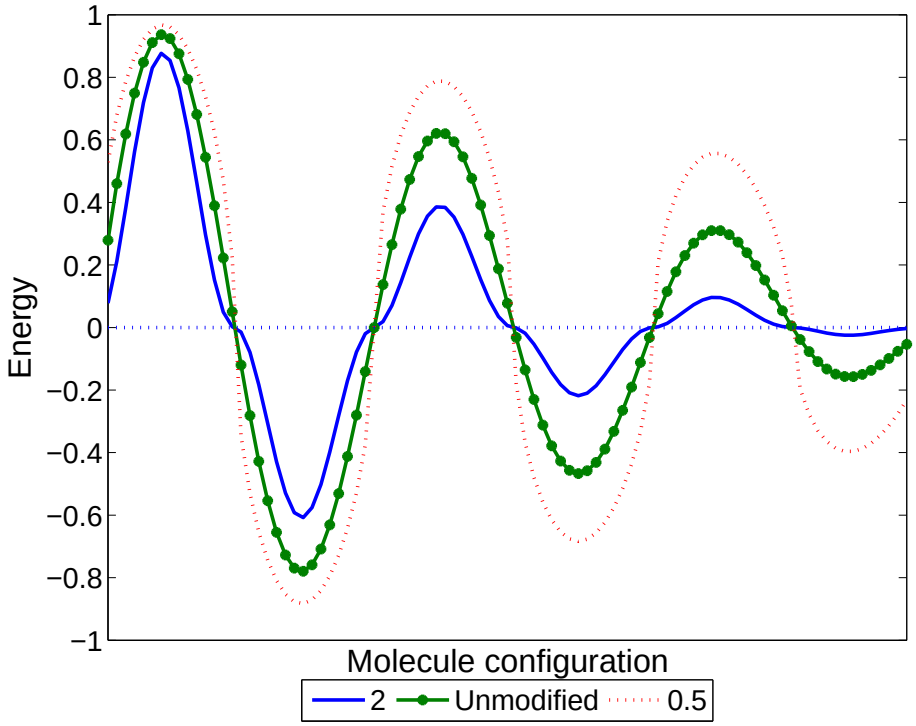

Figure 4.2: Comparison of unmodified energy landscape with $n = 1$, XEL with $n = 2$, and XEL with $n = 0.5$ in the range of $|E| < 1$. The $n = 0.5$ XEL accentuates the extrema while the $n = 2$ XEL flattens the landscape.

the XEL. These algorithms are implemented in simulations and are compared to unmodified algorithms.

4.1.2 Newton's Method with Exponential Energy Landscaping (Newton-XEL)

The Newton-XEL is derived from the Taylor's series expansion of the first derivative of the modified energy landscape E^*. Let h^* be a step that gives a minimum energy or zero gradient of the modified energy landscape. The Taylor series expansion of the derivative of the modified energy function becomes

$$0 = \frac{\partial E^*(\theta_k + h^*)}{\partial \theta} = \frac{\partial E^*(\theta_k)}{\partial \theta} + \frac{\partial^2 E^*(\theta_k)}{\partial^2 \theta} h^* + \frac{1}{2}\, h^{*\,T}\, \frac{\partial^3 E^*(\theta_k)}{\partial^3 \theta} h^* + \cdots$$

Dropping the high-order terms and rearranging gives the Newton-XEL step

$$\hat{h}^* = -\left(\frac{\partial^2 E^*(\theta_k)}{\partial^2 \theta}\right)^{-1} \frac{\partial E^*(\theta_k)}{\partial \theta} \tag{4.2}$$

where $\hat{h}^*$ is an estimation for h^*.

Next the first and the second derivatives of E^* are determined. Since

$$f(\theta) = E^* = sign(E)\,|E|^n = E\,|E|^{n-1} = E\,\sqrt{E^2}^{\,n-1}$$

the first derivative becomes

$$\begin{aligned}
\frac{\partial f(\theta)}{\partial \theta} = \frac{\partial E^*}{\partial \theta} &= \frac{\partial E}{\partial \theta}\,\sqrt{E^2}^{\,n-1} + E\frac{\partial\left(\sqrt{E^2}^{\,n-1}\right)}{\partial \theta} \\
&= -\tau\,\sqrt{E^2}^{\,n-1} + E(n-1)\,\sqrt{E^2}^{\,n-2}\left(\frac{1}{2\sqrt{E^2}}\right) 2E(-\tau) \\
&= -\left(\sqrt{E^2}^{\,n-1} + (n-1)\,\sqrt{E^2}^{\,n-1}\right)\tau \\
&= -n\,\sqrt{E^2}^{\,n-1}\,\tau \\
&= -n\,|E|^{n-1}\,\tau \\
&= -\beta\tau \\
&= -\tau^*
\end{aligned} \tag{4.3}$$

where $\beta \equiv n\,|E|^{n-1}$ and $\tau^* \equiv \beta\tau$ is a torque vector of E^*. The second derivative becomes

$$\begin{aligned}
\frac{\partial^2 f(\theta)}{\partial\theta^2} = \frac{\partial^2 E^*}{\partial\theta^2} &= -n\left(\frac{\partial\left(\sqrt{E^2}^{n-1}\right)}{\partial\theta}\tau + \sqrt{E^2}^{n-1}\frac{\partial\tau}{\partial\theta}\right) \\
&= -n\left(-(n-1)\sqrt{E^2}^{n-3}E(\tau\tau^T) + \sqrt{E^2}^{n-1}(-K)\right) \\
&= n\sqrt{E^2}^{n-1}\left((n-1)\frac{E}{\sqrt{E^2}^2}\tau\tau^T + K\right) \\
&= n\,|E|^{n-1}\left((n-1)\frac{E}{E^2}\tau\tau^T + K\right) \\
&= \beta\left((n-1)E^{-1}\tau\tau^T + K\right) \\
&\equiv K^*
\end{aligned} \tag{4.4}$$

where K^* is the Hessian matrix of E^*.

Substituting the first and the second derivative terms into Equation (4.2) the Newton-XEL step becomes

$$\hat{h}^* = (K^*)^{-1}\tau^* \tag{4.5}$$

Thus K^{-1} in Newton's method (3.5) is generalized to $(K^*)^{-1}$ which changes the direction and the magnitude of τ^* to give $\hat{h}^*$. Note that for $n = 1$, τ^* becomes τ, $(K^*)^{-1}$ becomes K^{-1}, and $\hat{h}^*$ becomes $\hat{h}$. While K^{-1} and $(K^*)^{-1}$ yield different magnitudes of $\hat{h}$ and $\hat{h}^*$, they unexpectedly yield the same direction. This characteristic and its proof are discussed in more detail in Section 4.2.2 in which it is also shown that the directions are distinct for the QNA and the QNA-XEL cases. The QNA-XEL is discusses below. Figure 4.3 gives a pseudo-code for the Newton-XEL with a line search.

Pseudo-code: Newton-XEL with a line search

Initialize ϵ, n, θ_0, and θ_1

Calculate E_0

for $k = 1, \ldots, k_{max}$ **do**

Calculate E_k, τ_k, and K_k

if $\left| \dfrac{E_k - E_{k-1}}{E_{k-1}} \right| < \epsilon$ **then**

`break` // Convergence criterion met.

end if

$\beta_k = n \left| E_k \right|^{n-1}$

$K_k^* = \beta_k \left((n-1) E_k^{-1} \tau_k \tau_k^T + K_k \right)$

$\tau_k^* = \beta_k \tau_k$

$\hat{h}_k^* = \left(K_k^* \right)^{-1} \tau_k^*$ // Determine the Newton-XEL step

$\alpha_k = \texttt{line\_search}(\hat{h}_k^*)$ // Determine the line search gain

$\theta_{k+1} = \theta_k + \alpha_k \hat{h}_k^*$ // $(k+1)^{\text{th}}$ solution

end for

Figure 4.3: A pseudo-code for Newton-XEL with a line search. Note that for Newton-XEL (*Hessian only*), described in Section 4.1.4, $\tau_k^* = \tau_k$.

4.1.3 Quasi-Newton Algorithm with Exponential Energy Landscaping (QNA-XEL)

The QNA-XEL is similar to the Newton-XEL update except that the Hessian matrix K is estimated. The QNA-XEL step is

$$\begin{aligned} \hat{h}^* &= \beta^{-1} \left((n-1) E^{-1} \tau \tau^T + \hat{K} \right)^{-1} \beta \tau \\ &= \left(\hat{K}^* \right)^{-1} \tau^* \end{aligned} \tag{4.6}$$

where

$$\hat{K}^* \equiv \beta \left((n-1) E^{-1} \tau \tau^T + \hat{K} \right) \tag{4.7}$$

is the estimated Hessian matrix of the modified energy landscape. Note that when $n = 1$ the QNA-XEL step (4.6) becomes the QNA step (3.6). Broyden's method (3.7) can be used to estimate $\hat{K}$ in the $\hat{K}^*$ but it reduces the effect of XEL on the QNA because it does not allow the direction of the QNA-XEL step to differ from the QNA step which is discussed in Section 4.2.2. The Broyden-XEL described below is used to estimate $\hat{K}^*$ instead.

4.1.3.1 Broyden's method with XEL (Broyden-XEL)

The Broyden-XEL has a different $\Delta\tau$ term defined by replacing E in Equation (3.7) with E^*,

$$\begin{aligned}\Delta\tau^*_{k-1} &= -\frac{\partial E^*(\theta_k)}{\partial\theta} - \left(-\frac{\partial E^*(\theta_{k-1})}{\partial\theta}\right) \\ &= \tau^*_k - \tau^*_{k-1}\end{aligned} \tag{4.8}$$

where $\Delta\tau^*$ is a change in the torque vector of E^*. Since $\tau^* = \beta\tau$, terms in $\Delta\tau^*$ are similar to $\Delta\tau$ but weighted by β's of the current and previous configurations, which contain the information about the energy landscape. In addition to $\Delta\tau^*$, the other two modified terms are $\hat{K}^*$ and $\hat{h}^*$ which are substituted into (3.7) to yield the Broyden-XEL update for $\hat{K}^*$

$$\hat{K}^*_k = \hat{K}^*_{k-1} + \left(\hat{h}^*_{k-1}{}^T \hat{h}^*_{k-1}\right)^{-1} \left(\Delta\tau^*_{k-1} - \hat{K}^*_{k-1}\hat{h}^*_{k-1}\right) \hat{h}^*_{k-1}{}^T \tag{4.9}$$

Note when $n = 1$ (4.9) reduces to (3.7).

As discussed in Section 4.2.2, the direction of the QNA-XEL step is different from that of the QNA step. From this point forward QNA-XEL is (4.6) that uses Broyden-XEL (4.9) to estimated $\hat{K}^*$ and QNA-XEL (*Broyden's method*) is (4.6) that uses Broyden's method (3.7) to estimate $\hat{K}$. Figure 4.4 gives a pseudo-code for the QNA-XEL with a line search.

Pseudo-code: QNA-XEL with a line search

Initialize ϵ, n, θ_0, θ_1, and $\hat{K}_0^*$

Calculate E_0, and τ_0

for $k = 1, \ldots, k_{max}$ **do**

Calculate E_k, and τ_k

if $\left| \dfrac{E_k - E_{k-1}}{E_{k-1}} \right| < \epsilon$ **then**

`break` // Convergence criterion met.

end if

Calculate $\hat{K}_k^*$

$$
\begin{aligned}
\hat{h}_{k-1}^* &= \theta_k - \theta_{k-1} \\
\beta_k &= n \cdot |E_k|^{n-1} \\
\beta_{k-1} &= n \cdot |E_{k-1}|^{n-1} \\
\Delta\tau_{k-1}^* &= \beta_k \tau_k - \beta_{k-1}\tau_{k-1} \\
\hat{K}_k^* &= \hat{K}_{k-1}^* + \left({\hat{h}_{k-1}^*}^T \hat{h}_{k-1}^* \right)^{-1} \left(\Delta\tau_{k-1}^* - \hat{K}_{k-1}^* \hat{h}_{k-1}^* \right) {\hat{h}_{k-1}^*}^T \\
\tau_k^* &= \beta_k \tau_k
\end{aligned}
$$

$\hat{h}_k^* = \left(\hat{K}_k^* \right)^{-1} \tau_k^*$ // Determine the QNA-XEL step

$\alpha_k =$ `line_search`$(\hat{h}_k^*)$ // Determine the line search gain

$\theta_{k+1} = \theta_k + \alpha_k \hat{h}_k^*$ // $(k+1)^{\text{th}}$ solution

end for

Figure 4.4: A pseudo-code for QNA-XEL with a line search. Note that for QNA-XEL (*Hessian only*), described in Section 4.1.4, $\tau_k^* = \tau_k$.

4.1.4 Broyden-Fletcher-Goldfarb-Shanno Algorithm with Exponential Energy Landscaping(BFGS-XEL)

Implementing XEL in BFGS changes the objective function from E to E^* and therefore τ becomes τ^*. The BFGS-XEL step becomes

$$\hat{h}_k^* = \hat{H}_k^* \tau_k^* \tag{4.10}$$

where $\hat{H}^*$ is the estimated inverse of the Hessian matrix of E^*. Substituting the terms for E^*, $\Delta\tau^*$ (4.8) and $\hat{h}^*$ into (3.8) yields the BFGS-XEL update for $\hat{H}^*$

$$\begin{aligned}\hat{H}_k^* = \hat{H}_{k-1}^* &- \left(\frac{\hat{H}_{k-1}^* \Delta\tau_{k-1}^* \, \hat{h}_{k-1}^{*\,T} + \hat{h}_{k-1}^* \, \Delta\tau_{k-1}^{*\,T} \, \hat{H}_{k-1}^*}{\hat{h}_{k-1}^{*\,T} \Delta\tau_{k-1}^*} \right) \\ &+ \left(\frac{\Delta\tau_{k-1}^{*\,T} \, \hat{H}_{k-1}^* \Delta\tau_{k-1}^*}{\hat{h}_{k-1}^{*\,T} \Delta\tau_{k-1}^*} - 1 \right) \frac{\hat{h}_{k-1}^* \, \hat{h}_{k-1}^{*\,T}}{\hat{h}_{k-1}^{*\,T} \Delta\tau_{k-1}^*}\end{aligned} \tag{4.11}$$

Used with a line search the BFGS-XEL update becomes

$$\begin{aligned}\theta_{k+1} &= \theta_k + \alpha_k \hat{H}_k^* \tau_k^* \\ &= \theta_k + \alpha_k \beta_k \hat{H}_k^* \tau_k\end{aligned}$$

since $\tau^* = \beta\tau$. Define $\alpha_k^* \equiv \alpha_k \beta_k$ is the new gain found by a line search without an explicit value of β_k. In theory α_k and α_k^* would globally minimize the energy function and yield the same total steps since $\hat{H}_k^* \tau_k^*$ and $\hat{H}_k^* \tau_k$ have the same direction. However, in practice α_k and α_k^* only locally minimizes the energy function so they could yield different total steps since $\hat{H}_k^* \tau_k^*$ and $\hat{H}_k^* \tau_k$ have different magnitudes.

To study an impact of the different magnitudes of the step on the quality of resulted configurations and speed, a variation of the BFGS-XEL of which the step equals $\hat{H}_k^* \tau_k$ is investigated. Letting $\tau^* = \tau$ keeps the same notation and the variation

of the BFGS-XEL step (4.10) becomes

$$\begin{aligned}
\tau_k^* &= \tau_k \\
\hat{h}_k^* &= \hat{H}_k^* \tau_k^* \\
\alpha_k^* &= \texttt{line_search}(\hat{h}_k^*) \\
\theta_{k+1} &= \theta_k + \alpha_k^* \hat{h}_k^*
\end{aligned}$$

Letting $\tau^* = \tau$ is basically substituting only the estimated inverse of the Hessian matrix for E^*. From this point forward the variation of the BFGS-XEL is referred to as BFGS-XEL (*Hessian only*). Similarly, by letting $\tau_k^* = \tau_k$, the Newton-XEL and the QNA-XEL become Newton-XEL (*Hessian only*) and QNA-XEL (*Hessian only*). Note that at the end of this chapter Tables 4.2 through 4.5 on pages 83–86 summarize all presented methods.

Comparing the Newton-XEL,

$$\begin{aligned}
\hat{h}^* &= (K_k^*)^{-1} \tau_k^* \\
&= \beta^{-1} \left((n-1) E^{-1} \tau_k \tau_k^T + K \right)^{-1} \beta \tau_k \\
&= \left((n-1) E^{-1} \tau_k \tau_k^T + K \right)^{-1} \tau_k
\end{aligned} \tag{4.12}$$

to the Newton-XEL (*Hessian only*),

$$\begin{aligned}
\hat{h}^* &= (K_k^*)^{-1} \tau_k^* \\
&= \beta^{-1} \left((n-1) E^{-1} \tau_k \tau_k^T + K \right)^{-1} \tau_k
\end{aligned}$$

The Newton-XEL (*Hessian only*) contains an extra term β^{-1} which increases the influence of n on the magnitudes of the step. Since $\beta = n\,|E|^{n-1}$, $\beta^{-1} = \dfrac{1}{n}\,|E|^{1-n}$ which is inversely dependent on n so the magnitudes of the Newton-XEL step (*Hessian only*) are expected to be larger when $n < 1$ and smaller when $n > 1$. This is consistent with the influence of n on the magnitudes of the XEL step discussed in Section 4.2.4 and the simulations discussed in Section 5.

Figure 4.5 gives a pseudo-code for the BFGS-XEL with a line search.

Pseudo-code: BFGS-XEL with a line search

Initialize ϵ, n, θ_0, θ_1, and $\hat{H}_0^*$

Calculate E_0 and τ_0

for $k = 1, \ldots, k_{max}$ **do**

Calculate E_k, and τ_k

if $\left| \dfrac{E_k - E_{k-1}}{E_{k-1}} \right| < \epsilon$ **then**

`break` // Convergence criterion met.

end if

Calculate $\hat{H}_k^*$

$$
\begin{aligned}
\hat{h}_{k-1}^* &= \theta_k - \theta_{k-1} \\
\beta_k &= n \cdot |E_k|^{n-1} \\
\beta_{k-1} &= n \cdot |E_{k-1}|^{n-1} \\
\Delta\tau_{k-1}^* &= \beta_k \tau_k - \beta_{k-1} \tau_{k-1} \\
\hat{H}_k^* &= \hat{H}_{k-1}^* - \left(\frac{\hat{H}_{k-1}^* \Delta\tau_{k-1}^* \, \hat{h}_{k-1}^{*\,T} + \hat{h}_{k-1}^* \, \Delta\tau_{k-1}^{*\,T} \hat{H}_{k-1}^*}{\hat{h}_{k-1}^{*\,T} \Delta\tau_{k-1}^*} \right) \\
&\quad + \left(\frac{\Delta\tau_{k-1}^{*\,T} \hat{H}_{k-1}^* \Delta\tau_{k-1}^*}{\hat{h}_{k-1}^{*\,T} \Delta\tau_{k-1}^*} - 1 \right) \frac{\hat{h}_{k-1}^* \, \hat{h}_{k-1}^{*\,T}}{\hat{h}_{k-1}^{*\,T} \Delta\tau_{k-1}^*} \\
\tau_k^* &= \beta_k \tau_k
\end{aligned}
$$

$\hat{h}_k^* = \hat{H}_k^* \tau_k^*$ // Determine the BFGS-XEL step

$\alpha_k = \texttt{line\_search}(\hat{h}_k^*)$ // Determine the line search gain

$\theta_{k+1} = \theta_k + \alpha_k \hat{h}_k^*$ // $(k+1)^{\text{th}}$ solution

end for

Figure 4.5: A pseudo-code for BFGS-XEL with a line search. Note that for BFGS-XEL (*Hessian only*) $\tau_k^* = \tau_k$.

4.2 Discussion on XEL

This section compares the Newton-XEL to the Newton steps with a basic assumption that these steps exist, i.e., K and K^* are invertible. Section 4.2.1 presents a sufficient condition for the positive definite Hessian matrix of the Newton-XEL step which is important for minimization. Section 4.2.2 shows that the Newton-XEL and the Newton steps are parallel and only the magnitude of Newton-XEL step is dependent on n. Section 4.2.4 shows that the magnitude of the Newton-XEL step is inversely related to n. Section 4.2.5 finds that the error residues from both algorithms are the same order. Section 4.2.6 discusses how the XEL method weights terms in the Hessian matrix and its estimation. Lastly, Section 4.2.7 concludes the discussions and summarizes the list of guidelines for changing n.

4.2.1 A Positive Definite Property of the Newton-XEL Hessian Matrix

Derived from the quadratic approximation of a function, Newton's method assumes that the function is convex with a global minimum if the Hessian matrix is positive definite and the Newton step gives the solution of the quadratic approximation. Although the actual function may not be convex and the Newton step may not give a local solution to the actual function, the positive definite property of the Newton Hessian matrix K implies that the Newton step is in descending direction.

This section shows that if the Hessian matrix of the Newton's method K is positive definite then so is K^* (the Hessian matrix of the Newton-XEL method) when an additional condition is satisfied. A positive definite K satisfies

$$x^T K x > 0; \quad \forall x \neq 0, x \in \Re^N$$

where N is the dimension of the energy landscape. Since $K^T = K$, it follows from

$$K^* = \beta \left((n-1) E^{-1} \tau \tau^T + K \right) \qquad [4.4]$$

that $K^* = K^{*T}$ and is thus symmetric. K^* is positive definite if

$$x^T (K^*) x > 0; \quad \forall x \neq 0, x \in \Re^N \tag{4.13}$$

Substituting in (4.4) and expanding gives

$$\beta(n-1)E^{-1} \left(\tau^T x\right)^T \tau^T x + \beta x^T K x > 0$$

Noting that $\beta = n\,|E|^{n-1} > 0$ because $n > 0$, that $\tau^T x$ is a scalar so

$$\left(\tau^T x\right)^T \tau^T x = \left|\tau^T x\right|^2 \geq 0$$

and that $x^T K x > 0$ then Equation (4.13) is satisfied when

$$(n-1)E^{-1} \geq 0$$

Note that this is only a sufficient condition for K^* to be positive definite. Further, since $n \neq 1$ corresponds to the modified energy landscape, $(n-1)E^{-1} \neq 0$ and the equality is dropped. Also, E^{-1} is undefined when $E = 0$ so the sufficient condition is stated as,

$$(n-1)E^{-1} > 0, \quad E \neq 0 \tag{4.14}$$

This condition can be satisfied in either of two ways:

$$E > 0 \;\text{ and }\; n > 1 \;\text{ or} \tag{4.15a}$$

$$E < 0 \;\text{ and }\; n < 1 \tag{4.15b}$$

For an adaptively modified energy landscape algorithm, Equation (4.15) gives a guideline for determining n so that K^* is positive definite. During the beginning of the conformation prediction process the energy is high and often positive, but as it progresses the energy decreases and often becomes negative. This suggests that $n > 1$ should be used at the beginning and $n < 1$ should be used toward the end of the process. However, enforcing the rule may not be very beneficial for the Newton-XEL method because it is derived under an assumption that K is positive definite so if K is not positive definite, K^* may not be.

Similarly it can be shown that the same condition applies to the estimated Hessian matrices of the QNA and the QNA-XEL (*Broyden's method*). With the same above reasoning, the Hessian matrix of the QNA-XEL (*Broyden's method*) $\hat{K}^*$ is positive definite when the Hessian matrix $\hat{K}$ is positive definite and Equation (4.15) is satisfied. However, enforcing the rule may not be very beneficial for the QNA-XEL (*Broyden's method*) because the actual K may not be positive definite and thus the QNA-XEL (*Broyden's method*) step may not be in descending direction. A more effective way to ensure the descending direction is using a line search which has been discussed in Section 3.1.1.

4.2.2 Direction of Newton-XEL Step

This section discusses the effect of XEL on the direction of the Newton-XEL step. Expanding the Newton-XEL step gives

$$\hat{h}^* = \left((n-1)E^{-1}\tau\tau^T + K\right)^{-1}\tau \qquad [4.12]$$

With the term $(n-1)E^{-1}\tau\tau^T$ added to K, the Newton-XEL step would be expected to be in different direction from the Newton step. However, only the magnitudes differ so they are actually parallel. Section 4.2.2.1 gives a derivation of the Newton-XEL step for a two dimensional (2D) energy landscape which is shown equal to the Newton step multiplied by a constant. Section 4.2.2.2 gives a proof for the N dimensional case.

4.2.2.1 Derivation of the 2D Newton-XEL step

Consider the K^* term in the Newton-XEL step. Since

$$\begin{aligned} K^* &= \beta\left((n-1)E^{-1}\tau\tau^T + K\right) \\ &= \begin{bmatrix} \beta\left(\frac{n-1}{E}\left(\frac{\partial E}{\partial\theta_1}\right)^2 + \frac{\partial^2 E}{\partial\theta_1{}^2}\right) & \beta\left(\frac{n-1}{E}\frac{\partial E}{\partial\theta_1}\frac{\partial E}{\partial\theta_2} + \frac{\partial^2 E}{\partial\theta_1\partial\theta_2}\right) \\ \beta\left(\frac{n-1}{E}\frac{\partial E}{\partial\theta_1}\frac{\partial E}{\partial\theta_2} + \frac{\partial^2 E}{\partial\theta_1\partial\theta_2}\right) & \beta\left(\frac{n-1}{E}\left(\frac{\partial E}{\partial\theta_2}\right)^2 + \frac{\partial^2 E}{\partial\theta_2{}^2}\right) \end{bmatrix} \end{aligned}$$

the inverse is,

$$(K^*)^{-1} = \frac{1}{\det(K^*)} \cdot \begin{bmatrix} \beta\left(\frac{n-1}{E}\left(\frac{\partial E}{\partial \theta_2}\right)^2 + \frac{\partial^2 E}{\partial \theta_2^2}\right) & \beta\left(-\frac{n-1}{E}\frac{\partial E}{\partial \theta_1}\frac{\partial E}{\partial \theta_2} - \frac{\partial^2 E}{\partial \theta_1 \partial \theta_2}\right) \\ \beta\left(-\frac{n-1}{E}\frac{\partial E}{\partial \theta_1}\frac{\partial E}{\partial \theta_2} - \frac{\partial^2 E}{\partial \theta_1 \partial \theta_2}\right) & \beta\left(\frac{n-1}{E}\left(\frac{\partial E}{\partial \theta_1}\right)^2 + \frac{\partial^2 E}{\partial \theta_1^2}\right) \end{bmatrix}$$

The Newton-XEL step is

$$\begin{aligned} \hat{h}^* &= (K^*)^{-1}\tau^* \\ &= \frac{1}{\det(K^*)} \cdot \begin{bmatrix} \beta^2\left(\frac{\partial^2 E}{\partial \theta_2^2}\left(-\frac{\partial E}{\partial \theta_1}\right) - \frac{\partial^2 E}{\partial \theta_1 \partial \theta_2}\left(-\frac{\partial E}{\partial \theta_2}\right)\right) \\ \beta^2\left(-\frac{\partial^2 E}{\partial \theta_1 \partial \theta_2}\left(-\frac{\partial E}{\partial \theta_1}\right) + \frac{\partial^2 E}{\partial \theta_1^2}\left(-\frac{\partial E}{\partial \theta_2}\right)\right) \end{bmatrix} \end{aligned}$$

and factorizing gives

$$\hat{h}^* = \frac{\beta^2}{\det(K^*)} \cdot \begin{bmatrix} \frac{\partial^2 E}{\partial \theta_2^2} & -\frac{\partial^2 E}{\partial \theta_1 \partial \theta_2} \\ -\frac{\partial^2 E}{\partial \theta_1 \partial \theta_2} & \frac{\partial^2 E}{\partial \theta_1^2} \end{bmatrix} \begin{bmatrix} -\frac{\partial E}{\partial \theta_1} \\ -\frac{\partial E}{\partial \theta_2} \end{bmatrix}$$

Since

$$\det(K) \cdot K^{-1} = \begin{bmatrix} \frac{\partial^2 E}{\partial \theta_2^2} & -\frac{\partial^2 E}{\partial \theta_1 \partial \theta_2} \\ -\frac{\partial^2 E}{\partial \theta_1 \partial \theta_2} & \frac{\partial^2 E}{\partial \theta_1^2} \end{bmatrix}$$

and

$$\tau = \begin{bmatrix} -\frac{\partial E}{\partial \theta_1} \\ -\frac{\partial E}{\partial \theta_2} \end{bmatrix}$$

then

$$\begin{aligned} \hat{h}^* &= \frac{\beta^2 \det(K)}{\det(K^*)} K^{-1}\tau \\ &= \frac{\beta^2 \det(K)}{\det(K^*)} \hat{h} \end{aligned} \tag{4.16}$$

Since the constant term $\frac{\beta^2 \det(K)}{\det(K^*)}$ only affects the length of the vector $K^{-1}\tau$, the direction of $\hat{h}^*$ only depends on K^{-1} not n and the Newton-XEL step is parallel to the Newton step. This means that XEL can have less effect on Newton's method than on other algorithms and that changing the n value may not significantly impact the Newton-XEL optimization path. Results presented in Chapter 5 show that on a special energy landscape case the Newton-XEL paths are not varied by modifying n at all.

Affecting only the magnitude of the Newton step, XEL acts similarly to a line search which finds a gain that is a scalar multiplier to the Newton step. One difference is that the line search goes through multiple iterations to find an "optimal" gain that minimizes the energy along the Newton step while the Newton-XEL only goes through one iteration and may not give an optimal gain. Thus, XEL can not replace a line search but can be used to improve performance of the Newton's method and a line search. The Newton-XEL with a line search may give the same results as the Newton's method with a line search, but not always, especially on a rough energy landscape. This is because these methods give different magnitudes of the step and therefore a line search can result in different gains corresponding to different local minima.

Similar to the Newton-XEL step, the QNA-XEL (*Broyden's method*) step, (4.6) and (3.7), on the 2D energy landscape can be derived to show that its direction depends on only the estimated Hessian matrix $\hat{K}$, i.e., $\hat{h}^* = \frac{\beta^2 \det(\hat{K})}{\det(\hat{K}^*)} \hat{K}^{-1}\tau$. Therefore, it has the same direction as the QNA step, (3.6) and (3.7), and it is independent of n.

The QNA-XEL uses the Broyden-XEL (4.9) which is dependent on n to estimate $\hat{K}^*$. To show that the direction of the QNA-XEL step is dependent on n, the 2D case

of the k^{th} QNA-XEL step is derived. Consider $\hat{K}_k^*$

$$\begin{aligned}\hat{K}_k^* &= \hat{K}_{k-1}^* + \left(\hat{h}_{k-1}^{*\,T}\hat{h}_{k-1}^*\right)^{-1}\left(\Delta\tau_{k-1}^* - \hat{K}_{k-1}^*\hat{h}_{k-1}^*\right)\hat{h}_{k-1}^{*\,T} \\ &= \tilde{K} + \Delta\tau_{k-1}^*\tilde{h}\end{aligned}$$

where $\tilde{K} \equiv \hat{K}_{k-1}^* - \left(\hat{h}_{k-1}^{*\,T}\hat{h}_{k-1}^*\right)^{-1}\hat{K}_{k-1}^*\left(\hat{h}_{k-1}^*\,\hat{h}_{k-1}^{*\,T}\right)$ with elements $\tilde{K}_i$, $i = 1, \cdots, 4$, and $\tilde{h} \equiv \left(\hat{h}_{k-1}^{*\,T}\hat{h}_{k-1}^*\right)^{-1}\hat{h}_{k-1}^{*\,T}$ with elements $\tilde{h}_i$, $i = 1, 2$, $\Delta\tilde{\tau}_i$ be elements of $\Delta\tau_{k-1}^*$. $\hat{K}_k^{*\,-1}$ can be written as

$$\hat{K}_k^{*\,-1} = \frac{1}{\det\left(\hat{K}_k^*\right)}\begin{bmatrix} \tilde{K}_4 + \Delta\tilde{\tau}_2\tilde{h}_2 & -\tilde{K}_2 - \Delta\tilde{\tau}_1\tilde{h}_2 \\ -\tilde{K}_3 - \Delta\tilde{\tau}_2\tilde{h}_1 & \tilde{K}_1 + \Delta\tilde{\tau}_1\tilde{h}_1 \end{bmatrix}$$

Let $\tilde{\tau}_i$ be elements of τ_k^* the QNA-XEL step becomes

$$\hat{K}_k^{*\,-1}\tau_k^* = \frac{1}{\det\left(\hat{K}_k^*\right)}\begin{bmatrix} \tilde{K}_4\tilde{\tau}_1 + \Delta\tilde{\tau}_2\tilde{h}_2\tilde{\tau}_1 - \tilde{K}_2\tilde{\tau}_2 - \Delta\tilde{\tau}_1\tilde{h}_2\tilde{\tau}_2 \\ -\tilde{K}_3\tilde{\tau}_1 - \Delta\tilde{\tau}_2\tilde{h}_1\tilde{\tau}_1 + \tilde{K}_1\tilde{\tau}_2 + \Delta\tilde{\tau}_1\tilde{h}_1\tilde{\tau}_2 \end{bmatrix}$$

$$= \frac{1}{\det\left(\hat{K}_k^*\right)}\begin{bmatrix} \tilde{K}_4\tilde{\tau}_1 - \tilde{K}_2\tilde{\tau}_2 + \left(\Delta\tilde{\tau}_2\tilde{\tau}_1 - \Delta\tilde{\tau}_1\tilde{\tau}_2\right)\tilde{h}_2 \\ -\tilde{K}_3\tilde{\tau}_1 + \tilde{K}_1\tilde{\tau}_2 - \left(\Delta\tilde{\tau}_2\tilde{\tau}_1 - \Delta\tilde{\tau}_1\tilde{\tau}_2\right)\tilde{h}_1 \end{bmatrix}$$

Since the scalar $\Delta\tilde{\tau}_2\tilde{\tau}_1 - \Delta\tilde{\tau}_1\tilde{\tau}_2$ which depends on n can not be factored out and it does not equal zero, the direction of the QNA-XEL step is dependent of n. This is expected to be true for the N dimensional case.

4.2.2.2 N dimensional Newton-XEL step

For an N dimensional energy landscape the Newton-XEL step is shown to always be parallel to the Newton step so the value of n does not affect the direction of the Newton-XEL step.

Proof that the Newton-XEL step is parallel to the Newton step:

Let $\hat{h}$ and $\hat{h}^*$ be the Newton and the Newton-XEL steps of the N dimensional energy landscape respectively,

$$\hat{h} = K^{-1}\tau \qquad [3.5]$$

$$\hat{h}^* = (K^*)^{-1}\tau^* \qquad [4.5]$$
$$= \beta\,(K^*)^{-1}\tau$$

Eliminating τ gives

$$\hat{h}^* = \beta\,(K^*)^{-1}K\hat{h} \qquad (4.17)$$

Using K^* in (4.4) and then (3.5) gives

$$\begin{aligned}\hat{h}^* &= \left((n-1)E^{-1}\tau\tau^T + K\right)^{-1}K\hat{h}\\ &= \left((n-1)E^{-1}\left(K^{-1}\tau\right)\tau^T + I\right)^{-1}\hat{h}\\ &= \left((n-1)E^{-1}\hat{h}\tau^T + I\right)^{-1}\hat{h}\end{aligned}$$

However, $\hat{h}$ is an eigenvector of $\left((n-1)E^{-1}\hat{h}\tau^T + I\right)^{-1}$, and thus also its inverse so,

$$\begin{aligned}\left((n-1)E^{-1}\hat{h}\tau^T + I\right)\hat{h} &= (n-1)E^{-1}\hat{h}\tau^T\hat{h} + \hat{h}\\ &= \lambda\hat{h}\end{aligned}$$

where

$$\lambda = (n-1)E^{-1}(\tau^T\hat{h}) + 1 \qquad (4.18)$$

and where τ and $\hat{h}$ are column vectors and $\tau^T\hat{h}$ is a scalar. Thus

$$\hat{h}^* = \lambda\hat{h} \qquad (4.19)$$

and $\hat{h}^*$ is a scalar multiple of $\hat{h}$ so the Newton-XEL step is parallel to the Newton step. □

Clearly when $n = 1$, so the energy landscape is not modified, then $\lambda = 1$ and the Newton-XEL step becomes the Newton step. The scalar multiplier λ has several properties discussed next in Section 4.2.3.

4.2.3 Properties of the Scalar Multiplier λ

This section discusses several properties of λ, which are used to derive the relationship between n and the magnitude of Newton-XEL step in Section 4.2.4 and the error residue in the Newton-XEL method in Section 4.2.5. As Section 4.2.2.2 previously defined λ as an eigenvalue of a matrix, here Section 4.2.3.1 shows that λ is also an eigenvalue of another matrix $\beta K\left(K^*\right)^{-1}$. Section 4.2.3.2 derives the other eigenvalue of $\beta K\left(K^*\right)^{-1}$. Lastly, Section 4.2.3.3 discusses the relationship between λ and a positive definite property of K and K^*.

4.2.3.1 Eigenvalue λ

Here λ is found to be an eigenvalue of $\beta K\left(K^*\right)^{-1}$. Since

$$\hat{h}^* = \lambda \hat{h}$$

substituting $\hat{h}^*$ and $\hat{h}$ from (4.5) and (3.5) and multiplying by K yield

$$\beta K\left(K^*\right)^{-1} \tau = \lambda \tau$$

and thus τ is an eigenvector of $\beta K\left(K^*\right)^{-1}$ corresponding to an eigenvalue λ.

4.2.3.2 Other eigenvalues

This section finds other eigenvalues of $\beta K\left(K^*\right)^{-1}$ first for the 2D case and then a proof is given for the N-dimensional case. From the derivation of the 2D Newton-XEL step (4.16) the λ for this case is

$$\lambda = \frac{\beta^2 \det(K)}{\det\left(K^*\right)}$$

By the property of the determinant of a matrix product [67] this is equal to

$$\lambda = \beta^2 \det\left(K\left(K^*\right)^{-1}\right) = \det\left(\beta K\left(K^*\right)^{-1}\right)$$

Since the product of the eigenvalues equals the determinant of the matrix [67], the other eigenvalue of the 2D matrix $\beta K\left(K^*\right)^{-1}$ is 1. The same can be shown for

$\beta\left(K^*\right)^{-1}K$ since

$$\lambda = \det\left(\beta\left(K^*\right)^{-1}K\right)$$

The other eigenvalue of the 2D matrix $\beta\left(K^*\right)^{-1}K$ is also 1.

For the N-dimensional $\beta K\left(K^*\right)^{-1}$, the only additional eigenvalue is 1 with $N-1$ corresponding eigenvectors. This is shown in the following proof.

Proof that in N dimensions an eigenvalue 1 has $N-1$ corresponding eigenvectors: To show that 1 is an eigenvalue of $\beta K\left(K^*\right)^{-1}$ matrix, it is equivalent to show that 1 is an eigenvalue of the inverse matrix $\beta\left(K^*\right)K^{-1}$ with $N-1$ corresponding eigenvectors. For 1 to be an eigenvalue the following characteristic equation must be satisfied,

$$\det\left(\beta\left(K^*\right)K^{-1}-I\right) = \det\left(\beta\left((n-1)E^{-1}\tau\tau^T+K\right)K^{-1}-I\right) = 0$$

or

$$\det\left(\beta(n-1)E^{-1}\tau\tau^TK^{-1}\right) = 0$$

Determinant properties for the product of a matrix and a scalar [67] give

$$\left(\beta(n-1)E^{-1}\right)^N\det\left(\tau\tau^T\right)\det\left(K^{-1}\right) = 0$$

This is satisfied identically since $\tau\tau^T$ is a rank one matrix so $\det\left(\tau\tau^T\right) = 0$. Therefore, 1 is an eigenvalue of $\left(K^*\right)K^{-1}$.

To find a corresponding eigenvector to 1 it must satisfy

$$\begin{aligned}\left(\beta\left(K^*\right)K^{-1}-I\right)x &= 0\\ \left(\beta\left((n-1)E^{-1}\tau\tau^T+K\right)K^{-1}-I\right)x &= 0\\ \beta(n-1)E^{-1}\tau\tau^TK^{-1}x &= 0\end{aligned}$$

Let $y = K^{-1}x$ then

$$\left(\tau\tau^T\right)y = 0$$

so y is in the null space of $\left(\tau\tau^T\right)$ and has $N-1$ unique solutions. Since K has full rank x also has $N-1$ unique solutions which means the $\beta\left(K^*\right)K^{-1}$ matrix has $N-1$ eigenvectors corresponding to 1.

Hence, the $\beta K\left(K^*\right)^{-1}$ matrix also has 1 as the other eigenvalue with $N-1$ corresponding eigenvectors. □

A similar proof that matrix $\beta\left(K^*\right)^{-1}K$ has 1 as the eigenvalue with $N-1$ corresponding eigenvectors can be given as well.

4.2.3.3 A relationship between λ and a positive definite property

This section shows how λ is related to a positive definite property. Since the product of all eigenvalues is the determinant of the matrix,

$$\begin{aligned}\lambda &= \det\left(\beta K\left(K^*\right)^{-1}\right)\\ &= \frac{\beta^N \det(K)}{\det\left(K^*\right)}\\ &= \beta^N \prod_{i=1}^{N}\mu_i / \prod_{i=1}^{N}\mu_i^*\end{aligned} \tag{4.20}$$

where μ's are eigenvalues of K and μ^*'s are eigenvalues of K^*. Since a positive definite matrix has all positive eigenvalues, their product is positive. With β^N always positive, for both K and K^* to be positive definite λ must be positive. This is a necessary but not a sufficient condition because both $\det(K)$ and $\det\left(K^*\right)$ can be negative or either matrix can have an even number of negative eigenvalues. Note that Equation (4.20) gives a closed form of λ consistent with the 2D case.

4.2.4 Magnitude of Newton-XEL Step

This section derives another form of λ that reveals a relationship between the magnitude of the Newton-XEL step and n. From the property of eigenvectors and eigenvalues the inverse matrix $\beta^{-1}(K^*)K^{-1}$ has τ and $\frac{1}{\lambda}$ as an eigenvector and an eigenvalue,

$$\beta^{-1}(K^*)K^{-1}\tau = \frac{1}{\lambda}\tau$$

which reduces to

$$\left[(n-1)E^{-1}(\tau^T K^{-1}\tau) - \left(\frac{1}{\lambda} - 1\right)\right]\tau = 0$$

since $\tau^T K^{-1}\tau$ is a scalar. Vanishing of the scalar coefficient yields

$$\lambda = \frac{1}{1 + (n-1)E^{-1}(\tau^T K^{-1}\tau)} \tag{4.21}$$

Compared to Equation (4.18), this gives a clearer relationship between λ and n. While λ depends on the current properties of the energy landscape (E, τ, and K) it also has an inverse relationship with n. Because λ is undefined when the denominator is zero, the inverse relationship would be true on the continuous intervals of λ. Since the Newton-XEL step is directly related to λ (4.19), it has an inverse relationship with n. As n gets smaller the magnitude of the Newton-XEL step becomes larger and vice versa. Although it is not proven here, this relationship is expected to carry over to other Newtonian XEL methods (QNA-XEL and BFGS-XEL) and has been supported by simulation results.

Equation (4.21) can also confirm that for both K and K^* to be positive definite λ must be positive. When K is positive definite so is K^{-1} and thus $\tau^T K^{-1}\tau > 0$. Also, when both K and K^* are positive definite, then from (4.14) $(n-1)E^{-1} > 0$ and therefore $\lambda > 0$.

4.2.5 Error residue from Newton-XEL step

This section compares the error residue of the Newton-XEL step to that of the Newton step. They are found to be of the same order.

First, the error residue from Newton step is determined. Let h be a step from θ_k that gives a minimum energy or zero gradient of the unmodified energy landscape at $\theta_k + h$. The Taylor series expansion of the derivative of the unmodified energy function becomes

$$\begin{aligned} 0 &= \frac{\partial E(\theta_k + h)}{\partial \theta} \\ &= \frac{\partial E(\theta_k)}{\partial \theta} + \frac{\partial^2 E(\theta_k)}{\partial^2 \theta} h + \frac{1}{2} h^T \frac{\partial^3 E(\theta_k)}{\partial^3 \theta} h + O(h^3) \\ &= -\tau + Kh + \frac{1}{2} h^T \frac{\partial^3 E(\theta_k)}{\partial^3 \theta} h + O(h^3) \end{aligned} \tag{4.22}$$

The error residue ρ from using Newton step $\hat{h} = K^{-1}\tau$, which approximates h, is

$$\begin{aligned} \rho &= \frac{1}{2} \hat{h}^T \frac{\partial^3 E(\theta_k)}{\partial^3 \theta} \hat{h} + O(\hat{h}^3) \\ &= O(\hat{h}^{\,2}) \end{aligned} \tag{4.23}$$

Second, the error residue from Newton-XEL step is found. Let h^* be a step from θ_k that gives a minimum energy or zero gradient of the modified energy landscape at $\theta_k + h^*$. The Taylor series expansion of the derivatives of the modified energy function becomes

$$\begin{aligned} 0 &= \frac{\partial E^*(\theta_k + h^*)}{\partial \theta} \\ &= \frac{\partial E^*(\theta_k)}{\partial \theta} + \frac{\partial^2 E^*(\theta_k)}{\partial^2 \theta} h^* + \frac{1}{2}\, h^{*\,T}\, \frac{\partial^3 E^*(\theta_k)}{\partial^3 \theta} h^* + O(h^{*\,3}) \end{aligned} \tag{4.24}$$

Since the minima on the modified and the unmodified energy landscape are at the same locations, $\theta_k + h = \theta_k + h^*$ and therefore

$$h = h^*$$

Substituting h^* for h in Equation (4.22) yields

$$0 = -\tau + Kh^* + \frac{1}{2}\, h^{*\,T}\, \frac{\partial^3 E(\theta_k)}{\partial^3 \theta} h^* + O(h^{*\,3}) \tag{4.25}$$

In the energy landscaping approach, the Newton-XEL step is used to minimize the energy on the unmodified energy landscape. The error residue from using Newton-XEL step can be found by substituting the Newton-XEL step $\hat{h}^*$ for h^* in Equation (4.25). Since $\hat{h}^* = \lambda\hat{h}$ from Equation (4.19), the Newton-XEL error residue ρ^* becomes

$$\rho^* = -\tau + K\lambda\hat{h} + \frac{1}{2}\left(\lambda\hat{h}\right)^T \frac{\partial^3 E(\theta_k)}{\partial^3\theta}\lambda\hat{h} + O(\hat{h}^3)$$

$$\rho^* = -\tau + \lambda K\hat{h} + \frac{\lambda^2}{2}\hat{h}^T\frac{\partial^3 E(\theta_k)}{\partial^3\theta}\hat{h} + O(\hat{h}^3)$$

Substituting $\hat{h} = K^{-1}\tau$ in the second term gives

$$\rho^* = -(1-\lambda)\tau + \frac{\lambda^2}{2}\hat{h}^T\frac{\partial^3 E(\theta_k)}{\partial^3\theta}\hat{h} + O(\hat{h}^3) \tag{4.26}$$

Rearranging Equation (4.21) gives

$$1 - \lambda = \lambda(n-1)E^{-1}\left(\tau^T K^{-1}\tau\right)$$

which is substituted into Equation (4.26) to yield

$$\rho^* = -\lambda(n-1)E^{-1}\left(\tau^T K^{-1}\tau\right)\tau + \frac{\lambda^2}{2}\hat{h}^T\frac{\partial^3 E(\theta_k)}{\partial^3\theta}\hat{h} + O(\hat{h}^3)$$

Using $I = K^{-T}K^T$, the error residue becomes

$$\rho^* = -\lambda(n-1)E^{-1}\left(\tau^T K^{-T}K^T K^{-1}\tau\right)\tau + \frac{\lambda^2}{2}\hat{h}^T\frac{\partial^3 E(\theta_k)}{\partial^3\theta}\hat{h} + O(\hat{h}^3)$$

Since $\hat{h} = K^{-1}\tau$ and K is symmetric,

$$\begin{aligned}\rho^* &= -\lambda(n-1)E^{-1}\left(\hat{h}^T K\hat{h}\right)\tau + \frac{\lambda^2}{2}\hat{h}^T\frac{\partial^3 E(\theta_k)}{\partial^3\theta}\hat{h} + O(\hat{h}^3)\\ &= O(\hat{h}^2)\end{aligned} \tag{4.27}$$

The Newton-XEL error residue in Equation (4.27) is of the same order as the Newton error residue in Equation (4.23). Since the Newton-XEL method is Newton's method applied to the modified energy landscape, the convergence property of the

Newton-XEL should be the similar to that of Newton's method which is confirmed by the equal order of the error residues of both methods.

Because the error residue of Newton-XEL is directly dependent on λ, a bounded λ would encourage stable optimization and convergence. Therefore, a guideline for choosing n in an adaptively modified energy landscape is to satisfy

$$|\lambda| < \delta \tag{4.28}$$

where δ is a bounding constant. This guideline is expected to be applicable to the QNA-XEL and the BFGS-XEL as well.

4.2.6 Weights on Different Energy Landscape Information

This section discusses how the XEL method gives different weights to the different information on the energy landscape in the XEL Hessian matrices and their estimation.

*4.2.6.1 The Newton-XEL and the QNA-XEL (*Broyden's method*) Hessian matrices*

Consider the Newton step in Equation (3.5) and the Newton-XEL step in Equation (4.5).

$$\hat{h} = K^{-1}\tau \tag{3.5}$$

$$\begin{aligned} \hat{h}^* &= (K^*)^{-1}\,\tau^* \\ &= \left((n-1)E^{-1}\tau\tau^T + K\right)^{-1}\tau \end{aligned} \tag{4.5}$$

The term $(n-1)E^{-1}\tau\tau^T$ added to the Hessian matrix K contains the current energy value and gradient information of the energy landscape and the n value basically determines how much weight is given on this term. The higher the n value, the higher the weight.

Instead of K, the QNA-XEL (*Broyden's method*) method uses the estimated Hessian matrix $\hat{K}$, which is recursively estimated from the previous Hessian matrix and

the change of the energy gradients. Giving more weight to the $(n-1)E^{-1}\tau\tau^T$ term gives more weight to the current information. This helps when the estimation of K is not very accurate, which occurs at the beginning of the minimization or during the optimization of a bumpy energy landscape. Another good aspect of raising the weight or increasing n is that the step size is reduced. Forcing the optimization to move slower improves the accuracy of Hessian matrix estimation because the Hessian matrix does not vary as much from step to step. This is supported by simulation results from the Newton-XEL, the QNA-XEL, and the BFGS-XEL as they show that when $n > 1$ the quality of resulted configurations is better than when $n < 1$.

4.2.6.2 The QNA-XEL and the BFGS-XEL Hessian matrix and Hessian matrix inverse estimation

The value of n affects the calculation of the estimated Hessian matrix $\hat{K}^*$ and the estimated inverse of the Hessian matrix $\hat{H}^*$ in the QNA-XEL and the BFGS-XEL because $\Delta\tau^*_{k-1} = \beta_k\tau_k - \beta_{k-1}\tau_{k-1}$ (4.8) and $\beta = n\,|E|^{n-1}$. The terms β_k and β_{k-1} are weights between the current (k^{th} iteration) and the previous ($(k-1)^{\text{th}}$ iteration) information of the energy landscape. To give more weight to the current landscape information, n must be chosen so that $\beta_k > \beta_{k-1}$.

When $n > 1$ the exponent $n-1$ is positive and therefore $\beta_k > \beta_{k-1}$ if $|E_k| > |E_{k-1}| > 1$. This condition is satisfied when $E_k > E_{k-1} > 1$ or $E_k < E_{k-1} < -1$. Thus, at an increasing positive E or a decreasing negative E, $n > 1$ should be used. The opposite happens when $n < 1$; the exponent $n-1$ is negative and therefore $\beta_k > \beta_{k-1}$ if $|E_{k-1}| > |E_k| > 1$. This condition is satisfied when $E_{k-1} > E_k > 1$ or $E_{k-1} < E_k < -1$. Thus, at a decreasing positive E or an increasing negative E, $n < 1$ should be used.

The described weighting however has possible undesirable outcomes. First, it requires constant monitoring of the landscape condition which perhaps increases computational cost. Second, in a rugged energy landscape, it results in changing n back

and forth which may cause instability. Lastly, giving more weight to the current information with an increasing energy it rewards the optimization path to the non-descending direction.

To avoid these outcomes, another weighting gives more weight to a point with lower energy. This is, $\beta_k > \beta_{k-1}$ when $E_k < E_{k-1}$ or $\beta_k < \beta_{k-1}$ when $E_k > E_{k-1}$.

At the beginning of protein prediction simulation the energy values are higher and usually positive. If $1 < E_k < E_{k-1}$, $|E_k| < |E_{k-1}|$ and for $\beta_k > \beta_{k-1}$ the exponent $n-1$ must be negative so $n < 1$ should be used. This applies to the case of $1 < E_{k-1} < E_k$ as well. On the contrary, at the end of the simulation the energy values are lower and usually negative. If $E_k < E_{k-1} < -1$, $|E_k| > |E_{k-1}|$ and for $\beta_k > \beta_{k-1}$ the exponent $n-1$ must be positive so $n > 1$ should be used. This applies to the case of $E_{k-1} < E_k < -1$ as well.

To adaptively change the value of n, the effects of XEL on the Hessian matrix estimation suggest to use $n < 1$ at the beginning of the simulation ($E > 0$) and $n > 1$ near the end of the simulation ($E < 0$). This agrees with a previous assumption in Section 4.1 that $n < 1$ should be implemented during the beginning of the simulation because it flattens the landscape and $n > 1$ should be implemented near the end of the simulation because it makes the global minimum more prominent.

4.2.7 Conclusion for Discussion on XEL

Section 4.2 has discussed the effects of XEL on several quantities and has suggested guidelines for changing n. First, the positive definite property of the Newton-XEL Hessian matrix is discussed. The conditions of n and energy values that make the matrix positive definite are given. Then the Newton-XEL step is shown to be the scalar multiple of the Newton step ($\hat{h}^* = \lambda \hat{h}$). That is, they are parallel. Moreover, the scalar λ is an eigenvalue of a matrix $\beta K \left(K^*\right)^{-1}$ which leads to a proof that λ is inversely related to n. Next, the error residue of the Newton-XEL is found to be

of the same order as that of the Newton's method which shows that the Newton-XEL can have a similar convergence property as the Newton's method. The error also suggests that $|\lambda|$ should be bounded. Lastly, the XEL method is found to give different weights on terms in the Hessian matrix and its estimation.

Guidelines for adaptively modified energy landscaping suggested in Section 4.2 are summarized in Table 4.1. For Newton-XEL and QNA-XEL, Guideline *i.* gives conditions for obtaining a positive definite XEL Hessian matrix in different energy values (Section 4.2.1) and Guideline *ii.* gives conditions for increasing or decreasing the magnitude of the Newton-XEL step in any situation (Section 4.2.4). Guideline *ii.* is expected to be applicable to BFGS-XEL methods as well. Guideline *iii.* gives a condition for improving stability and convergence in any situation and it is expected to be applicable to all methods (Section 4.2.5). Guideline *iv.* gives a condition for improving quality when $\hat{K}$ is inaccurate (Section 4.2.6.1). Lastly, Guideline *v.* gives conditions for increasing weight on the lower energy point in different energy values (Section 4.2.6.2).

Table 4.1: Guidelines for adaptively modified energy landscaping.

Guideline	**For**	**Situation**	**Condition**	**Applied to**
i.	Positive definite XEL Hessian matrix	$E > 0$ $E < 0$	$n > 1$ $n < 1$	Newton-XEL QNA-XEL
ii.	Smaller $\lvert\hat{h}^*\rvert$ Larger $\lvert\hat{h}^*\rvert$	Any	Larger n Smaller n	All
iii.	Improving stability and convergence	Any	Bounded $\lvert\lambda\rvert$	All
iv.	Improving quality of configurations	Inaccurate $\hat{K}^*$ or $\hat{H}^*$	Larger n	QNA-XEL BFGS-XEL
v.	More weight on lower energy	$E < 0$ $E > 0$	$n > 1$ $n < 1$	QNA-XEL BFGS-XEL

Most guidelines are consistent with each other, but some are not. Guideline *iii.* suggests the magnitude of λ should be less than the bounding value, which can be achieved by increasing n according to Guideline *ii.* Doing so can improve the inaccurate Hessian matrix estimation, according to Guideline *iv.* Since Guidelines *i.* and *v.* give conflicting conditions, one of them will be applied at a time.

4.3 Adaptive Exponential Energy Landscaping (AXEL)

Since different values of the exponent variable n affect the XEL algorithms, n should be chosen to improve performance. Section 4.3.1 derives two optimization algorithms for AXEL: the adaptive-n and the varying-n XEL algorithms. The difficulty of implementing the adaptive-n XEL algorithms is discussed which leads to the conclusion that only the implementation of the varying-n XEL algorithms is feasible and therefore presented here. Section 4.3.2 introduces and categorizes AXEL schemes to be used with the varying-n XEL or the XEL algorithms depending on the type of scheme. Together the varying-n XEL or the XEL algorithms and the AXEL schemes compose the method of AXEL referred to simply as AXEL.

4.3.1 Optimization with AXEL

The AXEL step can be derived such that 1) it updates both dihedral angles and the n value or 2) it only updates the dihedral angles. In 1) the adaptive-n XEL algorithms simultaneously calculate the updates for dihedral angles and n. In 2) the update for n is determined separately by an AXEL scheme (discussed in Section 4.3.2), and then the varying-n XEL algorithms calculate the update for the dihedral angles with the new n. Both adaptive-n and varying-n XEL algorithms are derived for Newton's method, the QNA, and the BFGS algorithm.

4.3.1.1 Adaptive-n XEL algorithms

Adaptive-n Newton-XEL

When the energy landscape is adaptively modified, the function of the modified energy depends not only on the dihedral angles but also on the n value. The Taylor series expansion for the modified energy landscape about the independent variables θ and n is

$$E^*(\theta_k + h_\theta^*, n_k + h_n^*) = E^* + \left(\frac{\partial E^*}{\partial \theta}\right)^T h_\theta^* + \left(\frac{\partial E^*}{\partial n}\right)^T h_n^* + O\left(h_\theta^{*\,2}\right) + O\left(h_n^{*\,2}\right) \quad (4.29)$$

where h_θ^* and h_n^* are changes in dihedral angles and n respectively. Taking the partial derivative with respect to θ of both sides gives

$$\frac{\partial E^*(\theta_k + h_\theta^*, n_k + h_n^*)}{\partial \theta} = \frac{\partial E^*}{\partial \theta} + \frac{\partial^2 E^*}{\partial \theta^2} h_\theta^* + \frac{\partial^2 E^*}{\partial \theta \partial n} h_n^* + O\left(h_\theta^{*\,2}\right) + O\left(h_n^{*\,2}\right)$$

Once the higher order terms are dropped, the step for dihedral angles that approximates $\dfrac{\partial E^*(\theta_k + h_\theta^*, n_k + h_n^*)}{\partial \theta} = 0$ is

$$0 = \frac{\partial E^*}{\partial \theta} + \frac{\partial^2 E^*}{\partial \theta^2} \hat{h}_\theta^* + \frac{\partial^2 E^*}{\partial \theta \partial n} \hat{h}_n^* \quad (4.30)$$

$$\hat{h}_\theta^* = -\left(\frac{\partial^2 E^*}{\partial \theta^2}\right)^{-1} \left(\frac{\partial E^*}{\partial \theta} + \frac{\partial^2 E^*}{\partial \theta \partial n} \hat{h}_n^*\right) \quad (4.31)$$

where $\hat{h}_\theta^*$ is the step for dihedral angles, which is the approximation of h_θ^*.

Since Equation (4.30) contains two unknowns another equation is needed to simultaneously solve for $\hat{h}_\theta^*$ and $\hat{h}_n^*$. This is obtained from taking the partial derivative with respect to n of the Taylor series expansion for $E^*(\theta_k, n_k)$, Equation (4.29),

$$\frac{\partial E^*(\theta_k + h_\theta^*, n_k + h_n^*)}{\partial n} = \frac{\partial E^*}{\partial n} + \frac{\partial^2 E^*}{\partial n \partial \theta} h_\theta^* + \frac{\partial^2 E^*}{\partial n^2} h_n^* + O\left(h_\theta^{*\,2}\right) + O\left(h_n^{*\,2}\right)$$

Once the higher order terms are dropped, the step for the exponenet n that approximates $\dfrac{\partial E^*(\theta_k + h_\theta^*, n_k + h_n^*)}{\partial n} = 0$ is

$$0 = \frac{\partial E^*}{\partial n} + \frac{\partial^2 E^*}{\partial n \partial \theta} \hat{h}_\theta^* + \frac{\partial^2 E^*}{\partial n^2} \hat{h}_n^* \quad (4.32)$$

$$\hat{h}_n^* = -\left(\frac{\partial^2 E^*}{\partial n^2}\right)^{-1} \left(\frac{\partial E^*}{\partial n} + \frac{\partial^2 E^*}{\partial n \partial \theta} \hat{h}_\theta^*\right) \quad (4.33)$$

where $\hat{h}_n^*$ is the step for n, which is the approximation of h_n^*.

To solve for h_θ^* and h_n^* simultaneously, stacking and rearranging Equations (4.30) and (4.32) gives

$$-\begin{bmatrix} \dfrac{\partial E^*}{\partial \theta} \\ \dfrac{\partial E^*}{\partial n} \end{bmatrix} = \begin{bmatrix} \dfrac{\partial^2 E^*}{\partial \theta^2} & \dfrac{\partial^2 E^*}{\partial \theta \partial n} \\ \dfrac{\partial^2 E^*}{\partial n \partial \theta} & \dfrac{\partial^2 E^*}{\partial n^2} \end{bmatrix} \begin{bmatrix} \hat{h}_\theta^* \\ \hat{h}_n^* \end{bmatrix}$$

Defining

$$\Theta \equiv \begin{bmatrix} \theta \\ n \end{bmatrix}$$

$$\kappa^* \equiv \begin{bmatrix} \dfrac{\partial^2 E^*}{\partial \theta^2} & \dfrac{\partial^2 E^*}{\partial \theta \partial n} \\ \dfrac{\partial^2 E^*}{\partial n \partial \theta} & \dfrac{\partial^2 E^*}{\partial n^2} \end{bmatrix} \tag{4.34}$$

$$\frac{\partial E^*}{\partial \Theta} \equiv \begin{bmatrix} \dfrac{\partial E^*}{\partial \theta} \\ \dfrac{\partial E^*}{\partial n} \end{bmatrix} \tag{4.35}$$

gives the adaptive-n Newton algorithm on AXEL

$$\begin{aligned} -\frac{\partial E^*}{\partial \Theta} &= \kappa^* \hat{h}_\Theta^* \\ \hat{h}_\Theta^* &= -\left(\kappa^*\right)^{-1} \frac{\partial E^*}{\partial \Theta} \end{aligned} \tag{4.36}$$

where $\hat{h}_\Theta^* \equiv \begin{bmatrix} \hat{h}_\theta^* \\ \hat{h}_n^* \end{bmatrix}$ is the adaptive-n Newton step on AXEL.

The terms in the matrix κ^* and the vector $\dfrac{\partial E^*}{\partial \Theta}$ are found from Equations (4.3) and (4.4),

$$\frac{\partial E^*}{\partial \theta} = -\tau^* = -\beta\tau \tag{4.3}$$

and

$$\frac{\partial^2 E^*}{\partial \theta^2} = K^* = \beta\left((n-1)E^{-1}\tau\tau^T + K\right) \tag{4.4}$$

The term $\dfrac{\partial^2 E^*}{\partial n \partial \theta}$, which is equivalent to $\dfrac{\partial^2 E^*}{\partial \theta \partial n}$, is derived by taking the partial derivative of $\dfrac{\partial E^*}{\partial \theta}$ with respect to n,

$$\frac{\partial^2 E^*}{\partial n \partial \theta} = \frac{\partial}{\partial n}\frac{\partial E^*}{\partial \theta} = \frac{\partial(-\beta\tau)}{\partial n}$$

Since $\beta = n\,|E|^{n-1}$ and τ does not depend on n,

$$\begin{aligned}
\frac{\partial^2 E^*}{\partial n \partial \theta} &= -\frac{\partial(n\,|E|^{n-1})}{\partial n}\tau \\
&= -\left(|E|^{n-1} + n\frac{\partial(|E|^{n-1})}{\partial n}\right)\tau \\
&= -\left(|E|^{n-1} + n\,|E|^{n-1}\ln|E|\right)\tau \\
&= -\left(1 + n\ln|E|\right)|E|^{n-1}\,\tau
\end{aligned} \tag{4.37}$$

The terms $\dfrac{\partial E^*}{\partial n}$ and $\dfrac{\partial^2 E^*}{\partial n^2}$ follow as,

$$\begin{aligned}
\frac{\partial E^*}{\partial n} &= \frac{\partial\left(E\,|E|^{n-1}\right)}{\partial n} \\
&= E\cdot\ln|E|\cdot|E|^{n-1}
\end{aligned} \tag{4.38}$$

and

$$\begin{aligned}
\frac{\partial^2 E^*}{\partial n^2} &= \frac{\partial}{\partial n}\left(\frac{\partial E^*}{\partial n}\right) \\
&= \frac{\partial\left(E\cdot\ln|E|\cdot|E|^{n-1}\right)}{\partial n} \\
&= E\left(\ln|E|\right)^2|E|^{n-1}
\end{aligned} \tag{4.39}$$

Therefore, substituting Equations (4.4), (4.37), and (4.39) into (4.34) and (4.35) yields

$$\kappa^* = \begin{bmatrix} \beta\left((n-1)E^{-1}\tau\tau^T + K\right) & -\left(1+n\ln|E|\right)|E|^{n-1}\,\tau \\ -\left(1+n\ln|E|\right)|E|^{n-1}\,\tau^T & E\left(\ln|E|\right)^2|E|^{n-1} \end{bmatrix} \tag{4.40}$$

and

$$\frac{\partial E^*}{\partial \Theta} = \begin{bmatrix} -\beta\tau \\ E\cdot\ln|E|\cdot|E|^{n-1} \end{bmatrix} \tag{4.41}$$

Equations (4.36), (4.40), and (4.41) define the adaptive-n Newton-XEL step.

Adaptive-n QNA-XEL

Two different algorithms for adaptive-n QNA-XEL can be obtained by estimating two matrices that define the Hessian matrix. Similar to the adaptive-n Newton-XEL, the first adaptive-n QNA-XEL is given by

$$\hat{h}^*_\Theta = -\left(\hat{\kappa}^*\right)^{-1} \frac{\partial E^*}{\partial \Theta} \tag{4.42}$$

and

$$\hat{\kappa}^* = \begin{bmatrix} \beta\left((n-1)E^{-1}\tau\tau^T + \hat{K}\right) & -\left(1+n\ln|E|\right)|E|^{n-1}\,\tau \\ -\left(1+n\ln|E|\right)|E|^{n-1}\,\tau^T & E\left(\ln|E|\right)^2|E|^{n-1} \end{bmatrix} \tag{4.43}$$

where $\hat{\kappa}^*$ is an estimation of κ^* and $\hat{K}$ is an estimation of K which was previously described by the Broyden's method as

$$\hat{K}_k = \hat{K}_{k-1} + \left(\hat{h}^{*\,T}_{\theta\,k-1}\,\hat{h}^*_{\theta\,k-1}\right)^{-1}\left(\Delta\tau_{k-1} - \hat{K}_{k-1}\,\hat{h}^*_{\theta\,k-1}\right)\hat{h}^{*\,T}_{\theta\,k-1} \qquad [3.7]$$

Note that $\hat{h}^*_{\theta\,k}$ here is equal to $\hat{h}_k$ in Equation (3.7).

The Broyden-XEL Equation (4.9) can also be used to calculate $\hat{K}^*$ but the $\Delta\tau^*$ and β terms have to be similarly redefined,

$$\hat{K}^*_k = \hat{K}^*_{k-1} + \left(\hat{h}^{*\,T}_{\theta\,k-1}\,\hat{h}^*_{\theta\,k-1}\right)^{-1}\left(\Delta\tau^{*\,k}_{\;\;k-1} - \hat{K}^*_{k-1}\,\hat{h}^*_{\theta\,k-1}\right)\hat{h}^{*\,T}_{\theta\,k-1} \tag{4.44}$$

where

$$\Delta\tau^{*\,k}_{\;\;k-1} = \beta^k_k\tau_k - \beta^k_{k-1}\tau_{k-1}$$

$$\beta^k_k = n_k\,|E_k|^{n_k-1}$$

$$\beta^k_{k-1} = n_k\,|E_{k-1}|^{n_k-1}$$

This definition of β^k and $\Delta\tau^{*\,k}$ uses the current value of n as if n is constant. Allowing only changes in θ to affect the $\hat{K}$ calculation is appropriate since $\hat{K}$ is an estimation of the stiffness matrix that is only of function of θ.

Alternatively, the second adaptive-n QNA-XEL uses Broyden's method to estimate the entire κ^* matrix as

$$\hat{\kappa}^*_k = \hat{\kappa}^*_{k-1} - \left(\hat{h}^{*\,T}_{\Theta\,k}\, \hat{h}^*_{\Theta\,k} \right)^{-1} \left(\frac{\partial E^*\left(\Theta_k\right)}{\partial \Theta} - \frac{\partial E^*\left(\Theta_{k-1}\right)}{\partial \Theta} + \hat{\kappa}^*_{k-1}\, \hat{h}^*_{\Theta\,k} \right) \hat{h}^{*\,T}_{\Theta\,k} \tag{4.45}$$

However, estimating only $\hat{K}$ and calculating other terms individually may give a more accurate estimation of κ^*.

Adaptive-n BFGS-XEL

Similar to previous BFGS algorithms, the adaptive-n BFGS-XEL algorithm estimates $(\kappa^*)^{-1}$.

$$\begin{aligned} \hat{h}^*_{\Theta\,k} &= \hat{H}^\kappa_k \frac{\partial E^*\left(\Theta_k\right)}{\partial \Theta} \\ \gamma^*_{k-1} &= \frac{\partial E^*\left(\Theta_k\right)}{\partial \Theta} - \frac{\partial E^*\left(\Theta_{k-1}\right)}{\partial \Theta} \\ \hat{h}^*_{\Theta\,k-1} &= \Theta_k - \Theta_{k-1} \end{aligned} \tag{4.46}$$

$$\begin{aligned} \hat{H}^\kappa_k = \hat{H}^\kappa_{k-1} &- \left(\frac{\hat{H}^\kappa_{k-1} \gamma^*_{k-1}\, \hat{h}^{*\,T}_{\Theta\,k-1} + \hat{h}^*_{\Theta\,k-1}\, \gamma^{*\,T}_{k-1}\, \hat{H}^\kappa_{k-1}}{\hat{h}^{*\,T}_{\Theta\,k-1}\, \gamma^*_{k-1}} \right) \\ &+ \left(1 + \frac{\gamma^{*\,T}_{k-1}\, \hat{H}^\kappa_{k-1} \gamma^*_{k-1}}{\hat{h}^{*\,T}_{\Theta\,k-1}\, \gamma^*_{k-1}} \right) \frac{\hat{h}^*_{\Theta\,k-1}\, \hat{h}^{*\,T}_{\Theta\,k-1}}{\hat{h}^{*\,T}_{\Theta\,k-1}\, \gamma^*_{k-1}} \end{aligned} \tag{4.47}$$

where $\hat{H}^\kappa$ is the estimation of $(\kappa^*)^{-1}$.

Discussion on adaptive-n algorithms

In addition to the dihedral angles, all adaptive-n XEL algorithms introduce the variable n for optimization. This additional variable inevitably increases the complexity of the problem. Unlike the algorithms with a fixed n, the optimization in adaptive-n XEL algorithms is done on an adaptively modified energy landscape. This means the value of the modified energy or the cost function is driven by not only the dihedral angles but also by changes in n. In fact, the change in n can have a larger impact on the energy value than changes in the dihedral angles. For $n > 1$, increasing n

deepens the valleys on the energy landscape. Therefore, while the modified energy value appears to be reduced because of changing n, the dihedral angles may be varied only slightly, or not at all. This presents an optimization difficulty, which is seen in some preliminary simulations implementing these adaptive-n algorithms. Results from these simulations show that the optimization paths sometimes move with very small steps and occasionally diverge.

Although an adaptive algorithm for determining n value is desired, simultaneously optimizing n and dihedral angles may not be a good solution. The next section derives the varying-n XEL algorithms that determine the change in n and then separately determine the changes in the dihedral angles.

4.3.1.2 Varying-n XEL algorithms

Since the protein conformation prediction problem is already difficult, the strategic goal of the varying-n XEL algorithms is not to aggravate the complexity of the problem while taking into account the effect of the changing n. The n value is changed according to different AXEL schemes discussed in Section 4.3.2.

Varying-n Newton-XEL

Consider the step for dihedral angles that approximates $\dfrac{\partial E^*(\theta_k + h^*_\theta, n_k + h^*_n)}{\partial \theta} = 0$,

$$\hat{h}^*_\theta = -\left(\frac{\partial^2 E^*}{\partial \theta^2}\right)^{-1}\left(\frac{\partial E^*}{\partial \theta} + \frac{\partial^2 E^*}{\partial n \partial \theta}\hat{h}^*_n\right) \qquad [4.31]$$

Clearly, when n is constant or $\hat{h}^*_n = 0$, this step becomes the Newton-XEL step in Equation (4.5). Although E^* is now a function of variables n and θ, the locations of the minima where $\dfrac{\partial E^*}{\partial \theta} = 0$ remain the same. Therefore, $\hat{h}^*_\theta$ can be used to locate the minimum on the unmodified energy landscape $\dfrac{\partial E}{\partial \theta} = 0$.

Substituting all necessary terms into Equation (4.31) yields the varying-n Newton-XEL step,

$$\begin{aligned}
\hat{h}_\theta^* &= -\left(\beta\left((n-1)E^{-1}\tau\tau^T + K\right)\right)^{-1}\left(-\beta\tau - (1+n\ln|E|)\,|E|^{n-1}\,\tau\hat{h}_n^*\right) \\
&= \beta^{-1}\left((n-1)E^{-1}\tau\tau^T + K\right)^{-1}\cdot\beta\cdot\left(1+\left(\frac{1}{n}+\ln|E|\right)\hat{h}_n^*\right)\tau \\
&= \left((n-1)E^{-1}\tau\tau^T + K\right)^{-1}\left(1+\left(\frac{1}{n}+\ln|E|\right)\hat{h}_n^*\right)\tau \\
&= (K^*)^{-1}\left(1+\left(\frac{1}{n}+\ln|E|\right)\hat{h}_n^*\right)\tau^* \qquad (4.48)
\end{aligned}$$

Compared to the Newton-XEL step $\hat{h}^*$ (4.5), the varying-n Newton-XEL step in (4.48) has an additional term, $\left(\frac{1}{n}+\ln|E|\right)\hat{h}_n^*$, which reflects the change in the energy landscape due to the changing n. Since $1+\left(\frac{1}{n}+\ln|E|\right)\hat{h}_n^*$ is a scalar, the varying-n Newton-XEL step is a scalar multiple of the Newton-XEL step. Recalling Equation (4.19) gives

$$\begin{aligned}
\hat{h}_\theta^* &= \left(1+\left(\frac{1}{n}+\ln|E|\right)\hat{h}_n^*\right)\hat{h}^* \\
&= \left(1+\left(\frac{1}{n}+\ln|E|\right)\hat{h}_n^*\right)\lambda\hat{h} \\
&= \lambda^\Theta\hat{h}
\end{aligned}$$

where $\lambda^\Theta \equiv \left(1+\left(\frac{1}{n}+\ln|E|\right)\hat{h}_n^*\right)\lambda$ is a scalar multiplier of the Newton-XEL with Θ as an independent variable. Since the varying-n Newton-XEL step is a scalar multiple of the Newton step, its error residue can be shown to be of the same order as Newton error residue. Similar to a guideline that limits $|\lambda|$ (4.28), $\left|\lambda^\Theta\right| < \delta$ can be followed.

Varying-n QNA-XEL

Similarly, the varying-n QNA-XEL step is defined as

$$\begin{aligned}
\hat{h}_\theta^* &= \left((n-1)E^{-1}\tau\tau^T + \hat{K}\right)^{-1}\left(1+\left(\frac{1}{n}+\ln|E|\right)\hat{h}_n^*\right)\tau \\
&= \left(\hat{K}^*\right)^{-1}\left(1+\left(\frac{1}{n}+\ln|E|\right)\hat{h}_n^*\right)\tau^* \qquad (4.49)
\end{aligned}$$

where $\hat{K}^*$ is calculated using the redefined Broyden-XEL Equation (4.44). Again, the varying-n QNA-XEL step has the additional term $\left(\frac{1}{n} + \ln|E|\right)\hat{h}^*_n$ compared to the QNA-XEL step.

Varying-n BFGS-XEL

The varying-n BFGS-XEL step is defined as

$$\hat{h}^*_\theta = \hat{H}^* \left(1 + \left(\frac{1}{n} + \ln|E|\right)\hat{h}^*_n\right)\tau^* \tag{4.50}$$

where $\hat{H}^*$ is the estimation of $\left(\frac{\partial^2 E^*}{\partial\theta^2}\right)^{-1}$ previously defined in Equation (4.11). For the varying-n case the β and $\Delta\tau^*$ terms have to be redefined as,

$$\begin{aligned}
\hat{h}^*_{\theta\,k-1} &= \theta_k - \theta_{k-1} \\
\beta^k_k &= n_k \cdot |E_k|^{n_k - 1} \\
\beta^k_{k-1} &= n_k \cdot |E_{k-1}|^{n_k - 1} \\
\Delta\tau^{*\,k}_{\;\;k-1} &= \beta^k_k \tau_k - \beta^k_{k-1}\tau_{k-1}
\end{aligned}$$

$$\begin{aligned}
\hat{H}^*_k = \hat{H}^*_{k-1} &- \left(\frac{\hat{H}^*_{k-1}\,\Delta\tau^{*\,k}_{\;\;k-1}\,\hat{h}^{*\,T}_{\theta\,k-1} + \hat{h}^*_{\theta\,k-1}\,\Delta\tau^{*\,k}_{\;\;k-1}{}^{T}\,\hat{H}^*_{k-1}}{\hat{h}^{*\,T}_{\theta\,k-1}\,\Delta\tau^{*\,k}_{\;\;k-1}}\right) \\
&+ \left(\frac{\Delta\tau^{*\,k}_{\;\;k-1}{}^{T}\,\hat{H}^*_{k-1}\,\Delta\tau^{*\,k}_{\;\;k-1}}{\hat{h}^{*\,T}_{\theta\,k-1}\,\Delta\tau^{*\,k}_{\;\;k-1}} - 1\right)\frac{\hat{h}^*_{\theta\,k-1}\,\hat{h}^{*\,T}_{\theta\,k-1}}{\hat{h}^{*\,T}_{\theta\,k-1}\,\Delta\tau^{*\,k}_{\;\;k-1}}
\end{aligned} \tag{4.51}$$

Again, compared to the BFGS-XEL, the varying-n BFGS-XEL contains the additional term $\left(\frac{1}{n} + \ln|E|\right)\hat{h}^*_n$. It turns out that this term is very important to protein predictions that use a varying n during the minimization process. In fact, a comparative study shows that when n is not constant the varying-n BFGS-XEL can locate a conformation with up to 9% lower energy than that located by the BFGS-XEL.

Figures 4.6, 4.7, and 4.8 give pseudo-codes for the varying-n Newton-XEL, the varying-n QNA-XEL, and the varying-n BFGS-XEL, respectively, used with a line search and an AXEL scheme.

Pseudo-code: Varying-n Newton-XEL with a line search

Initialize ϵ, n_1, and θ_1

Calculate E_0

for $k = 1, \ldots, k_{max}$ **do**

Calculate E_k, τ_k, and K_k

if $\left| \dfrac{E_k - E_{k-1}}{E_{k-1}} \right| < \epsilon$ **then**

`break` // Convergence criterion met.

end if

$n_{k+1} =$ `AXEL_scheme`() // Determine the new exponent with an AXEL scheme

$\hat{h}^*_{n\,k} = n_{k+1} - n_k$

$\beta_k = n_k \left| E_k \right|^{n_k - 1}$

$K^*_k = \beta_k \left((n_k - 1) E_k^{-1} \tau_k \tau_k^T + K_k \right)$

$\tau^*_k = \beta_k \tau_k$

$\hat{h}^*_{\theta\,k} = (K^*_k)^{-1} \left(1 + \left(\dfrac{1}{n_k} + \ln \left| E_k \right| \right) \hat{h}^*_{n\,k} \right) \tau^*_k$

// Determine the varying-n Newton-XEL step

$\alpha_k =$ `line_search`$(\hat{h}^*_{\theta\,k})$ // Determine the line search gain

$\theta_{k+1} = \theta_k + \alpha_k \, \hat{h}^*_{\theta\,k}$ // $(k+1)^{\text{th}}$ solution

end for

Figure 4.6: A pseudo-code for varying-n Newton-XEL with a line search. Note that for varying-n Newton-XEL (*Hessian only*) $\tau^*_k = \tau_k$.

Pseudo-code: Varying-n QNA-XEL with a line search

Initialize ϵ, n_1, θ_0, θ_1, and $\hat{K}_0^*$

Calculate E_0, and τ_0

for $k = 1, \ldots, k_{max}$ **do**

Calculate E_k, and τ_k

if $\left| \dfrac{E_k - E_{k-1}}{E_{k-1}} \right| < \epsilon$ **then**

`break` // Convergence criterion met.

end if

$n_{k+1} =$ `AXEL_scheme`() // Determine the new exponent with an AXEL scheme

$\hat{h}^*_{n\,k} = n_{k+1} - n_k$

Calculate $\hat{K}_k^*$

$$\begin{aligned}
\hat{h}^*_{\theta\,k-1} &= \theta_k - \theta_{k-1} \\
\beta_k^k &= n_k \cdot |E_k|^{n_k - 1} \\
\beta_{k-1}^k &= n_k \cdot |E_{k-1}|^{n_k - 1} \\
\Delta {\tau^*}^{k}_{k-1} &= \beta_k^k \tau_k - \beta_{k-1}^k \tau_{k-1} \\
\hat{K}_k^* &= \hat{K}_{k-1}^* + \left({\hat{h}^*_{\theta\,k-1}}^T \hat{h}^*_{\theta\,k-1} \right)^{-1} \left(\Delta {\tau^*}^{k}_{k-1} - \hat{K}_{k-1}^* \hat{h}^*_{\theta\,k-1} \right) {\hat{h}^*_{\theta\,k-1}}^T
\end{aligned}$$

$\tau_k^* = \beta_k^k \tau_k$

$\hat{h}^*_{\theta\,k} = \left(\hat{K}_k^* \right)^{-1} \left(1 + \left(\dfrac{1}{n_k} + \ln |E_k| \right) \hat{h}^*_{n\,k} \right) \tau_k^*$

// Determine the varying-n QNA-XEL step

$\alpha_k =$ `line_search`$(\hat{h}^*_{\theta\,k})$ // Determine the line search gain

$\theta_{k+1} = \theta_k + \alpha_k \hat{h}^*_{\theta\,k}$ // $(k+1)^{\text{th}}$ solution

end for

Figure 4.7: A pseudo-code for varying-n QNA-XEL with a line search. Note that for varying-n QNA-XEL (*Hessian only*) $\tau_k^* = \tau_k$.

Pseudo-code: Varying-n BFGS-XEL with a line search

Initialize ϵ, n_1, θ_0, θ_1, and $\hat{H}_0^*$

Calculate E_0 and τ_0

for $k = 1, \ldots, k_{max}$ **do**

Calculate E_k, and τ_k

if $\left| \dfrac{E_k - E_{k-1}}{E_{k-1}} \right| < \epsilon$ **then**

`break` // Convergence criterion met.

end if

$n_{k+1} =$ `AXEL_scheme`() // Determine the exponent with an AXEL scheme

$\hat{h}^*_{n\,k} = n_{k+1} - n_k$

Calculate $\hat{H}_k^*$

$$
\begin{aligned}
\hat{h}^*_{\theta\,k-1} &= \theta_k - \theta_{k-1} \\
\beta_k^k &= n_k \cdot |E_k|^{n_k - 1} \\
\beta_{k-1}^k &= n_k \cdot |E_{k-1}|^{n_k - 1} \\
{\Delta\tau^*}_{k-1}^{\,k} &= \beta_k^k \tau_k - \beta_{k-1}^k \tau_{k-1} \\
\hat{H}_k^* = \hat{H}_{k-1}^* &- \left(\frac{\hat{H}_{k-1}^* \, {\Delta\tau^*}_{k-1}^{\,k} \, {\hat{h}^*_{\theta\,k-1}}^T + \hat{h}^*_{\theta\,k-1} \, {{\Delta\tau^*}_{k-1}^{\,k}}^T \hat{H}_{k-1}^*}{{\hat{h}^*_{\theta\,k-1}}^T \, {\Delta\tau^*}_{k-1}^{\,k}} \right) \\
&+ \left(\frac{{{\Delta\tau^*}_{k-1}^{\,k}}^T \hat{H}_{k-1}^* \, {\Delta\tau^*}_{k-1}^{\,k}}{{\hat{h}^*_{\theta\,k-1}}^T \, {\Delta\tau^*}_{k-1}^{\,k}} - 1 \right) \frac{\hat{h}^*_{\theta\,k-1} \, {\hat{h}^*_{\theta\,k-1}}^T}{{\hat{h}^*_{\theta\,k-1}}^T \, {\Delta\tau^*}_{k-1}^{\,k}}
\end{aligned}
$$

$\tau_k^* = \beta_k^k \tau_k$

$\hat{h}^*_{\theta\,k} = \hat{H}_k^* \left(1 + \left(\dfrac{1}{n_k} + \ln |E_k| \right) \hat{h}^*_{n\,k} \right) \tau_k^*$

// Determine the varying-n BFGS-XEL step

$\alpha_k =$ `line_search`$(\hat{h}^*_{\theta\,k})$ // Determine the line search gain

$\theta_{k+1} = \theta_k + \alpha_k \, \hat{h}^*_{\theta\,k}$ // $(k+1)^{\text{th}}$ solution

end for

Figure 4.8: A pseudo-code for varying-n BFGS-XEL with a line search. Note that for varying-n BFGS-XEL (*Hessian only*) $\tau_k^* = \tau_k$.

4.3.2 AXEL Schemes

The simulation of conformation prediction typically consists of two parts, probabilistic search and minimization. AXEL schemes can be categorized into two groups based on when n changes. Some of these schemes change n during the minimization, so they are applicable to varying-n XEL algorithms. Others change n less frequently outside the minimization stage and therefore they are more applicable to XEL algorithms.

I. Global: n changes depending on the global location of the probabilistic search which is the searched location with respect to the global solution. For example, as the search progresses the searched location is expected to be closer to the global solution so larger n may be needed to improve quality. These changes of n happen during the course of the simulation, but n's are constant during each minimization step. XEL algorithms are used in minimization.

- ***i.*** **Periodic:** n changes at specific times. The performance of the XEL simulations are compared to determine which n performs the best during the beginning, the middle, and the end of the simulation. The AXEL scheme changes the n value accordingly.

II. Local: n changes depending on the local location of the search. These changes happen during the course of each minimization step. Varying-n XEL algorithms are used in minimization.

- ***i.*** **Periodic:** n changes at specific times.
- ***ii.*** **Occasional:** n changes when criteria are met.

Global and local schemes are implemented so that the global and local effects of the AXEL on the prediction simulation can be studied. While global schemes are only applicable to a protein prediction algorithm that involves several minimization

steps for many starting configurations, local schemes are applicable to an algorithm with one minimization step as well.

4.4 Conclusion for the Energy Landscaping

The method of XEL has been presented and discussed, and five guidelines on how to change the landscape are suggested. The derivations of Newton's method, the QNA, and the BFGS method for XEL and AXEL have been presented. Lastly, the AXEL schemes used with the varying-n XEL algorithms are categorized. The method of adaptively modified energy landscape comprises of the varying-n XEL or the XEL algorithms and the AXEL schemes. Tables 4.2 through 4.5 summarize all presented methods from Chapters 3 and 4. Note that in some equations subscript k is omitted for displaying purposes.

Table 4.2: A summary of presented methods: Newton, Newton-XEL, QNA, and QNA-XEL (1 of 4)

Method	Equation
Newton	$\hat{h}_k = K_k^{-1}\tau_k$ [3.5]
Newton-XEL	$K_k^* = \beta_k\left((n-1)E_k^{-1}\tau_k\tau_k^T + K_k\right)$ [4.4] $\tau_k^* \equiv \beta_k\tau_k \quad$ (or $\tau_k^* \equiv \tau_k$, *Hessian only*) $\hat{h}_k^* = \left(K_k^*\right)^{-1}\tau_k^*$ [4.5]
QNA	$\hat{K}_k = \hat{K}_{k-1} + \left(\hat{h}_{k-1}^T\hat{h}_{k-1}\right)^{-1}$ $\cdot\left(\Delta\tau_{k-1} - \hat{K}_{k-1}\hat{h}_{k-1}\right)\hat{h}_{k-1}^T$ [3.7] $\hat{h}_k = \hat{K}_k^{-1}\tau_k$ [3.6]
QNA-XEL	$\hat{K}_k^* = \hat{K}_{k-1}^* + \left(\hat{h}_{k-1}^{*\,T}\hat{h}_{k-1}^*\right)^{-1}$ $\cdot\left(\Delta\tau_{k-1}^* - \hat{K}_{k-1}^*\hat{h}_{k-1}^*\right)\hat{h}_{k-1}^{*\,T}$ [4.9] $\tau_k^* \equiv \beta_k\tau_k \quad$ (or $\tau_k^* \equiv \tau_k$, *Hessian only*) $\hat{h}_k^* = \left(\hat{K}_k^*\right)^{-1}\tau_k^*$ [4.6]
QNA-XEL (*Broyden's method*)	$\hat{K}_k = \hat{K}_{k-1} + \left(\hat{h}_{k-1}^{*\,T}\hat{h}_{k-1}^*\right)^{-1}$ $\cdot\left(\Delta\tau_{k-1} - \hat{K}_{k-1}\,\hat{h}_{k-1}^*\right)\hat{h}_{k-1}^{*\,T}$ [3.7] $\hat{K}_k^* = \beta_k(n-1)E_k^{-1}\tau_k\tau_k^T + \hat{K}_k$ [4.7] $\hat{h}_k^* = \left(\hat{K}_k^*\right)^{-1}\tau_k^*$ [4.6]

Table 4.3: A summary of presented methods: BFGS and BFGS-XEL (2 of 4)

Method (cont.)	**Equation (cont.)**
BFGS	$\hat{H}_k = \hat{H}_{k-1} - \left(\frac{\hat{H}_{k-1}\Delta\tau_{k-1}\hat{h}_{k-1}^T + \hat{h}_{k-1}\Delta\tau_{k-1}^T\hat{H}_{k-1}}{\hat{h}_{k-1}^T\Delta\tau_{k-1}} \right) + \left(\frac{\Delta\tau_{k-1}^T\hat{H}_{k-1}\Delta\tau_{k-1}}{\hat{h}_{k-1}^T\Delta\tau_{k-1}} - 1 \right) \frac{\hat{h}_{k-1}\hat{h}_{k-1}^T}{\hat{h}_{k-1}^T\Delta\tau_{k-1}}$ [3.8]
	$\hat{h}_k = \hat{H}_k\tau_k$ [3.9]
BFGS-XEL	$\hat{H}_k^* = \hat{H}_{k-1}^* - \left(\frac{\hat{H}_{k-1}^*\Delta\tau_{k-1}^*\ \hat{h}_{k-1}^*{}^T + \hat{h}_{k-1}^*\ \Delta\tau_{k-1}^*{}^T\ \hat{H}_{k-1}^*}{\hat{h}_{k-1}^*{}^T\ \Delta\tau_{k-1}^*} \right)$ $+ \left(\frac{\Delta\tau_{k-1}^*{}^T\ \hat{H}_{k-1}^*\Delta\tau_{k-1}^*}{\hat{h}_{k-1}^*{}^T\ \Delta\tau_{k-1}^*} - 1 \right) \frac{\hat{h}_{k-1}^*\ \hat{h}_{k-1}^*{}^T}{\hat{h}_{k-1}^*{}^T\ \Delta\tau_{k-1}^*}$ [4.11]
	$\tau_k^* \equiv \beta_k\tau_k \qquad (\text{or } \tau_k^* \equiv \tau_k, \textit{ Hessian only})$
	$\hat{h}_k^* = \hat{H}_k^*\tau_k^*$ [4.10]

Table 4.4: A summary of presented methods: Adaptive-n XEL algorithms (3 of 4)

Method (cont.)	**Equation (cont.)**
Adaptive-n Newton-XEL	$\kappa^* = \begin{bmatrix} \beta\left((n-1)E^{-1}\tau\tau^T + K\right) & -\left(1+n\ln\lvert E\rvert\right)\lvert E\rvert^{n-1}\tau \\ -\left(1+n\ln\lvert E\rvert\right)\lvert E\rvert^{n-1}\tau^T & E\left(\ln\lvert E\rvert\right)^2\lvert E\rvert^{n-1} \end{bmatrix}$ [4.40] $\hat{h}^*_\Theta = -\left(\kappa^*\right)^{-1}\dfrac{\partial E^*}{\partial\Theta}$ [4.36]
Adaptive-n QNA-XEL *a*) Broyden's method for $\hat{K}$ *b*) Redefined Broyden-XEL *c*) Broyden's method for $\hat{\kappa}^*$	$\hat{\kappa}^* = \begin{bmatrix} \beta\left((n-1)E^{-1}\tau\tau^T + \hat{K}\right) & -\left(1+n\ln\lvert E\rvert\right)\lvert E\rvert^{n-1}\tau \\ -\left(1+n\ln\lvert E\rvert\right)\lvert E\rvert^{n-1}\tau^T & E\left(\ln\lvert E\rvert\right)^2\lvert E\rvert^{n-1} \end{bmatrix}$ [4.43] *a*) $\hat{K}_k = \hat{K}_{k-1} + \left(\hat{h}^{*T}_{\theta\,k-1}\,\hat{h}^*_{\theta\,k-1}\right)^{-1}\left(\Delta\tau_{k-1} - \hat{K}_{k-1}\,\hat{h}^*_{\theta\,k-1}\right)\hat{h}^{*T}_{\theta\,k-1}$ [3.7] *b*) $\hat{K}^*_k = \hat{K}^*_{k-1} + \left(\hat{h}^{*T}_{\theta\,k-1}\,\hat{h}^*_{\theta\,k-1}\right)^{-1} \cdot \left(\Delta\tau^{*\,k}_{\;\;k-1} - \hat{K}^*_{k-1}\,\hat{h}^*_{\theta\,k-1}\right)\hat{h}^{*T}_{\theta\,k-1}$ [4.44] *c*) $\hat{\kappa}^*_k = \hat{\kappa}^*_{k-1} - \left(\hat{h}^{*T}_{\Theta\,k}\,\hat{h}^*_{\Theta\,k}\right)^{-1} \left(\dfrac{\partial E^*\left(\Theta_k\right)}{\partial\Theta} - \dfrac{\partial E^*\left(\Theta_{k-1}\right)}{\partial\Theta} + \hat{\kappa}^*_{k-1}\,\hat{h}^*_{\Theta\,k}\right)\hat{h}^{*T}_{\Theta\,k}$ [4.45] $\hat{h}^*_\Theta = -\left(\hat{\kappa}^*\right)^{-1}\dfrac{\partial E^*}{\partial\Theta}$ [4.42]
Adaptive-n BFGS-XEL	$\hat{H}^\kappa_k = \hat{H}^\kappa_{k-1} - \left(\dfrac{\hat{H}^\kappa_{k-1}\gamma^*_{k-1}\,\hat{h}^{*T}_{\Theta\,k-1} + \hat{h}^*_{\Theta\,k-1}\,\gamma^{*T}_{k-1}\,\hat{H}^\kappa_{k-1}}{\hat{h}^{*T}_{\Theta\,k-1}\,\gamma^*_{k-1}}\right) + \left(1 + \dfrac{\gamma^{*T}_{k-1}\,\hat{H}^\kappa_{k-1}\gamma^*_{k-1}}{\hat{h}^{*T}_{\Theta\,k-1}\,\gamma^*_{k-1}}\right)\dfrac{\hat{h}^*_{\Theta\,k-1}\,\hat{h}^{*T}_{\Theta\,k-1}}{\hat{h}^{*T}_{\Theta\,k-1}\,\gamma^*_{k-1}}$ [4.47] $\hat{h}^*_{\Theta\,k} = \hat{H}^\kappa_k\dfrac{\partial E^*\left(\Theta_k\right)}{\partial\Theta}$ [4.46]
	where $\dfrac{\partial E^*}{\partial\Theta} = \begin{bmatrix} -\beta\tau \\ E\cdot\ln\lvert E\rvert\cdot\lvert E\rvert^{n-1} \end{bmatrix}$ [4.41]

Table 4.5: A summary of presented methods: Varying-n XEL algorithms (4 of 4)

Method (cont.)	**Equation (cont.)**	
Varying-n Newton-XEL	$\hat{h}^*_{\theta\,k} = (K^*_k)^{-1}\left(1+\left(\frac{1}{n_k}+\ln\lvert E_k\rvert\right)\hat{h}^*_{n\,k}\right)\tau^*_k$	[4.48]
Varying-n QNA-XEL *a*) Broyden's method for $\hat{K}$ *b*) Redefined Broyden-XEL	*a*) $\hat{K}_k = \hat{K}_{k-1} + \left(\hat{h}^{*\,T}_{\theta\,k-1}\,\hat{h}^*_{\theta\,k-1}\right)^{-1}\left(\Delta\tau_{k-1} - \hat{K}_{k-1}\,\hat{h}^*_{\theta\,k-1}\right)\hat{h}^{*\,T}_{\theta\,k-1}$	[3.7]
	$\hat{K}^*_k = \beta_k(n-1)E_k^{-1}\tau_k\tau_k^T + \hat{K}_k$	[4.7]
	b) $\hat{K}^*_k = \hat{K}^*_{k-1} + \left(\hat{h}^{*\,T}_{\theta\,k-1}\,\hat{h}^*_{\theta\,k-1}\right)^{-1}\left(\Delta\tau^{*\,k}_{\;\;k-1} - \hat{K}^*_{k-1}\,\hat{h}^*_{\theta\,k-1}\right)\hat{h}^{*\,T}_{\theta\,k-1}$	[4.44]
	$\hat{h}^*_{\theta\,k} = \left(\hat{K}^*_k\right)^{-1}\left(1+\left(\frac{1}{n_k}+\ln\lvert E_k\rvert\right)\hat{h}^*_{n\,k}\right)\tau^*_k$	[4.49]
Varying-n BFGS-XEL	$\hat{H}^*_k = \hat{H}^*_{k-1} - \left(\frac{\hat{H}^*_{k-1}\,\Delta\tau^{*\,k}_{\;\;k-1}\,\hat{h}^{*\,T}_{\theta\,k-1} + \hat{h}^*_{\theta\,k-1}\,\Delta\tau^{*\,k}_{\;\;k-1}{}^{T}\,\hat{H}^*_{k-1}}{\hat{h}^{*\,T}_{\theta\,k-1}\,\Delta\tau^{*\,k}_{\;\;k-1}}\right)$ $+\left(\frac{\Delta\tau^{*\,k}_{\;\;k-1}{}^{T}\,\hat{H}^*_{k-1}\,\Delta\tau^{*\,k}_{\;\;k-1}}{\hat{h}^{*\,T}_{\theta\,k-1}\,\Delta\tau^{*\,k}_{\;\;k-1}} - 1\right)\frac{\hat{h}^*_{\theta\,k-1}\,\hat{h}^{*\,T}_{\theta\,k-1}}{\hat{h}^{*\,T}_{\theta\,k-1}\,\Delta\tau^{*\,k}_{\;\;k-1}}$	[4.51]
	$\hat{h}^*_{\theta\,k} = \hat{H}^*_k\left(1+\left(\frac{1}{n_k}+\ln\lvert E_k\rvert\right)\hat{h}^*_{n\,k}\right)\tau^*_k$	[4.50]

CHAPTER V

SIMULATION RESULTS AND DISCUSSIONS

To investigate the effects of the method of exponential energy landscaping (XEL) on Newton's method, the quasi-Newton algorithm (QNA), and the Broyden-Fletcher-Goldfarb-Shanno (BFGS) algorithm, several simulations are performed with two different implementations.

(*i*) The investigated algorithms are implemented *without* a probabilistic search on a simulated two-dimensional energy landscape.

(*ii*) The investigated algorithms are implemented *with* a probabilistic search on a multi-dimensional energy landscape.

While the variations of the QNA-XEL and the BFGS-XEL algorithms are implemented in both **(*i*)** and **(*ii*)**, the variations of the Newton-XEL are only implemented in **(*i*)** because the second derivatives on a multi-dimensional energy landscape are not available in the platform used for **(*ii*)**. Observing the results from the simulations in **(*i*)**, adaptive exponential energy landscape (AXEL) schemes are developed and implemented in **(*ii*)**.

The performance comparison of different simulations is done in terms of quality and speed evaluated through different quantities for different implementations.

(*i*) For *quality* comparisons the converged energy and the number of paths that locate the global minimum are evaluated, and for *speed* comparisons the number of iterations is evaluated.

(*ii*) For *quality* comparisons the score improvement, the lowest score, and the similarity to the native conformation are evaluated, and for *speed* comparisons the

number of iterations is evaluated. In addition to quality and speed, efficiency defined as a ratio between quality and speed is evaluated.

Section 5.1 presents results from the following algorithms implemented in ***(i)***: Newton-XEL, Newton-XEL (*Hessian only*), QNA-XEL, QNA-XEL (*Hessian only*), BFGS-XEL, and BFGS-XEL (*Hessian only*). The XEL algorithms are derived from both the first derivative and the Hessian matrix of the XEL but the XEL (*Hessian only*) algorithms are derived from the Hessian matrix only. Section 5.2 presents results from the following algorithms implemented in ***(ii)***: QNA-XEL, QNA-XEL (*Hessian only*), BFGS-XEL, and BFGS-XEL (*Hessian only*). Section 5.3 discusses simulations and results from AXEL. Lastly, Section 5.4 concludes the chapter.

5.1 XEL Algorithms without Probabilistic Search

To study the effect of XEL on Newton's method, QNA, and BFGS, optimization paths are generated for each XEL algorithm on two simulated two-degree-of-freedom energy landscapes: simple and complex energy landscapes. As shown in Figure 5.1, the simple energy landscape (a) is a fourth order polynomial function with only one minimum which is a global minimum and the complex landscape (b) is a parabolic based surface with multiple local minima and one global minimum. The simple energy landscape and its derivatives as functions of the dihedral angles θ are described below. The functions of the complex energy landscape and its derivatives are not displayed because of their lengths.

$$
\begin{aligned}
E(\theta_1, \theta_2) &= 0.0128({\theta_1}^4 + {\theta_2}^4) - 16 \\
\frac{\partial E(\theta_1, \theta_2)}{\partial \theta} &= \begin{bmatrix} 0.0512\ {\theta_1}^3 \\ 0.0512\ {\theta_2}^3 \end{bmatrix} \\
\frac{\partial E^2(\theta_1, \theta_2)}{\partial \theta^2} &= \begin{bmatrix} 0.1536\ {\theta_1}^2 & 0 \\ 0 & 0.1536\ {\theta_2}^2 \end{bmatrix}
\end{aligned}
\tag{5.1}
$$

where E is the energy function and θ_i's are dihedral angles. Note that dihedral angles exist only on a protein chain or a polypeptide and the smallest polypeptide has two residues with four degrees of freedom. Therefore, there are no dihedral angles in 2D system but torsion angles. However, these angles are called dihedral angles here for consistency.

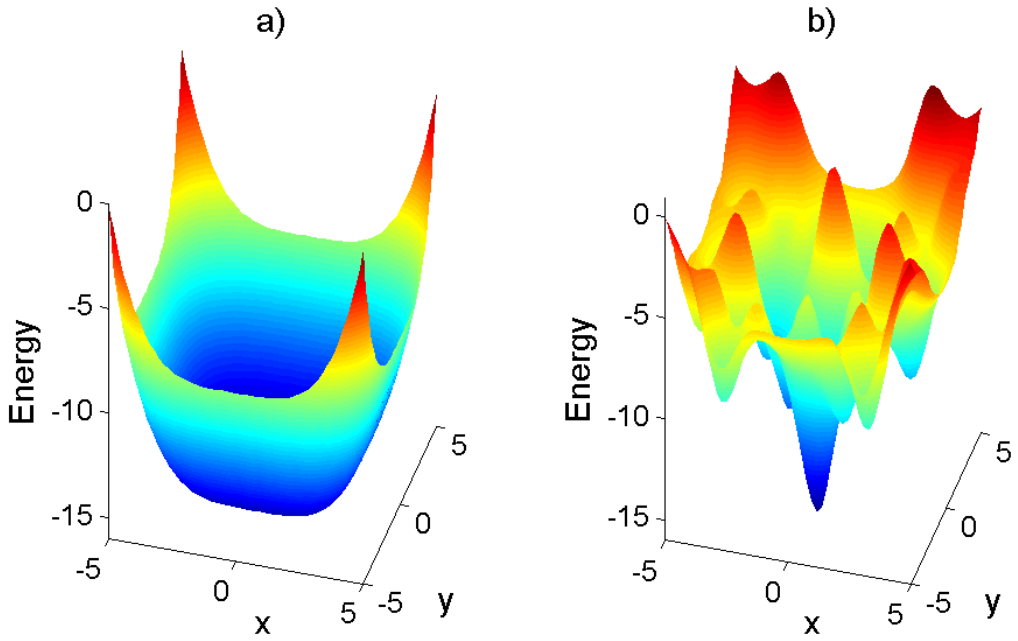

Figure 5.1: The simple energy landscape (a) and the complex landscape (b) used to generate optimization paths for all algorithms.

A grid search is done as paths are generated for 25 different locations equally distributed on the surface shown in Figure 5.2. Seven values of n used in these simulations are 0.125, 0.25, 0.5, 1 (unmodified case), 2, 3, and 4. The range of n values is selectively investigated because it sufficiently represents the performance of the algorithms. Also, as n becomes lower than 0.125 the optimization algorithm diverges and as n becomes higher than 4, total steps become too small to reach a solution in a reasonable time.

In addition to different optimization algorithms, two line search algorithms are used to demonstrate that a line search can have a significant impact to the XEL.

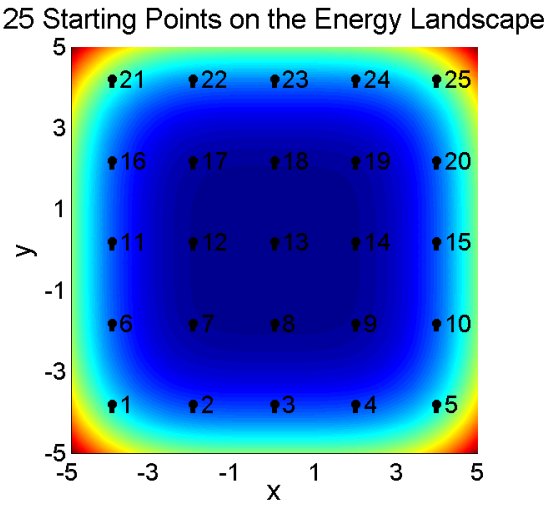

Figure 5.2: Twenty-five starting points for paths generation. Each line indicates the initial optimization step, which moves from the the dot to another end of the line.

The first implemented line search is an integrated golden section and quadratic interpolation bracketing algorithm with Brent's method [52]. This algorithm uses a bracketing algorithm combined with the golden section and quadratic interpolation to initialize a search interval and uses the Brent's method to locate the minimum. The second implemented line search is a heuristic exhaustive line search algorithm. This algorithm divides and checks the energies of a search interval and if an optimal line search gain can not be located it iteratively increases the range of the search up to five times the size of the computed step.

Two sets of the convergence criteria are used: 1) for the simple energy landscape the minimization stops when the magnitude of the gradient is less than 0.001, and 2) for the complex energy landscape the minimization stops when the magnitude of the gradient is less than 0.1. Since the simple energy landscape is very flat near the global minimum, the stricter criterion is in place so that it would allow the paths to reach the minimum. The magnitude of the gradient is chosen to be the measure of convergence because the usual measure, the relative change of the energy function between the current and the previous step, can mistakenly assume paths with a very small step as converged.

As mentioned, the performance comparison of different simulations is discussed in terms of quality and speed. Before the quality and speed are fully discussed, Section 5.1.1 discusses the characteristics of paths generated by the XEL algorithms with different values of n. Section 5.1.2 discusses the quality and speed of XEL algorithms by comparing the energy to which the paths converge and the number of iterations to the unmodified case. Section 5.1.3 compares the quality in terms of the ability of the algorithms to locate a global minimum.

5.1.1 Path Comparison

To study the effects of XEL on each algorithm, paths generated by Newton-XEL, Newton-XEL (*Hessian only*), QNA-XEL, QNA-XEL (*Hessian only*), BFGS-XEL, and BFGS-XEL (*Hessian only*) are evaluated. The equations of these algorithms are summarized in Tables 4.2 and 4.3, pages 83–84.

5.1.1.1 Newton-XEL and Newton-XEL (Hessian only)

Simple energy landscape

With a bracketing and Brent line search on the simple energy landscape, Newton-XEL and Newton-XEL (*Hessian only*) (4.5) paths, if exist, from all cases are the same as they are able to locate the minimum in one total step. This can be observed from Figures 5.3 and 5.4 with representative examples of paths from Newton-XEL and Newton-XEL (*Hessian only*) respectively. Note that Figures 5.3 c) and 5.4 c) do not have a path but only the starting points. Since n affects only the magnitude of the Newton step, as described in Section 4.2.2, and the Newton step of this energy landscape is coincidently parallel to a line from the current position to the global minimum, every Newton-XEL step always points to the global minimum. Although the magnitudes of these steps are different, the bracketing and Brent line search helps locating the minimum in one total step. If the Newton step does not always point to the global minimum, the Newton-XEL steps with different magnitudes would yield

different increments and more divergent paths of the Newton-XEL would have been observed. The absence of deviation in both Newton-XEL and Newton-XEL (*Hessian only*) paths demonstrates that the effect of XEL on the Newton's method can be very limited due to the lack of directional effects of XEL on the Newton's method.

Figures 5.3 c) and 5.4 c) show that at some starting points no path is generated. This is because at least one of the second derivatives (5.1) with respect to θ_1 or θ_2 is zero and therefore the Hessian matrix is not invertible at these starting points. This happens in 9 out of 25 starting points. This demonstrates one restriction of the Newton-XEL, which is inherited from the Newton's method.

With a heuristic exhaustive line search, Newton-XEL and Newton-XEL (*Hessian only*) paths on the simple energy landscape also make a straight line to the global minimum but with multiple total steps. This can be observed from Figures 5.5 and 5.6 with representative examples of paths from Newton-XEL and Newton-XEL (*Hessian only*) respectively. While the special characteristic of the energy landscape is that straight paths are taken to the minimum, most paths are actually composed of small total steps which are different from those with the bracketing and Brent line search that have only one total step. Because the heuristic exhaustive line search preserves the effect of n on the magnitude of the step, the line search results in different gains for different values of n.

With a heuristic exhaustive line search, most Newton-XEL and Newton-XEL (*Hessian only*) paths are the same except some $n = 0.125$ paths from Newton-XEL (*Hessian only*) that overshoot the minimum at the first total step and locate it soon thereafter (Figure 5.6 e). This is because with the Newton-XEL (*Hessian only*) n has a greater effect on the magnitude so the $n = 0.125$ total step becomes much larger than that of the Newton-XEL.

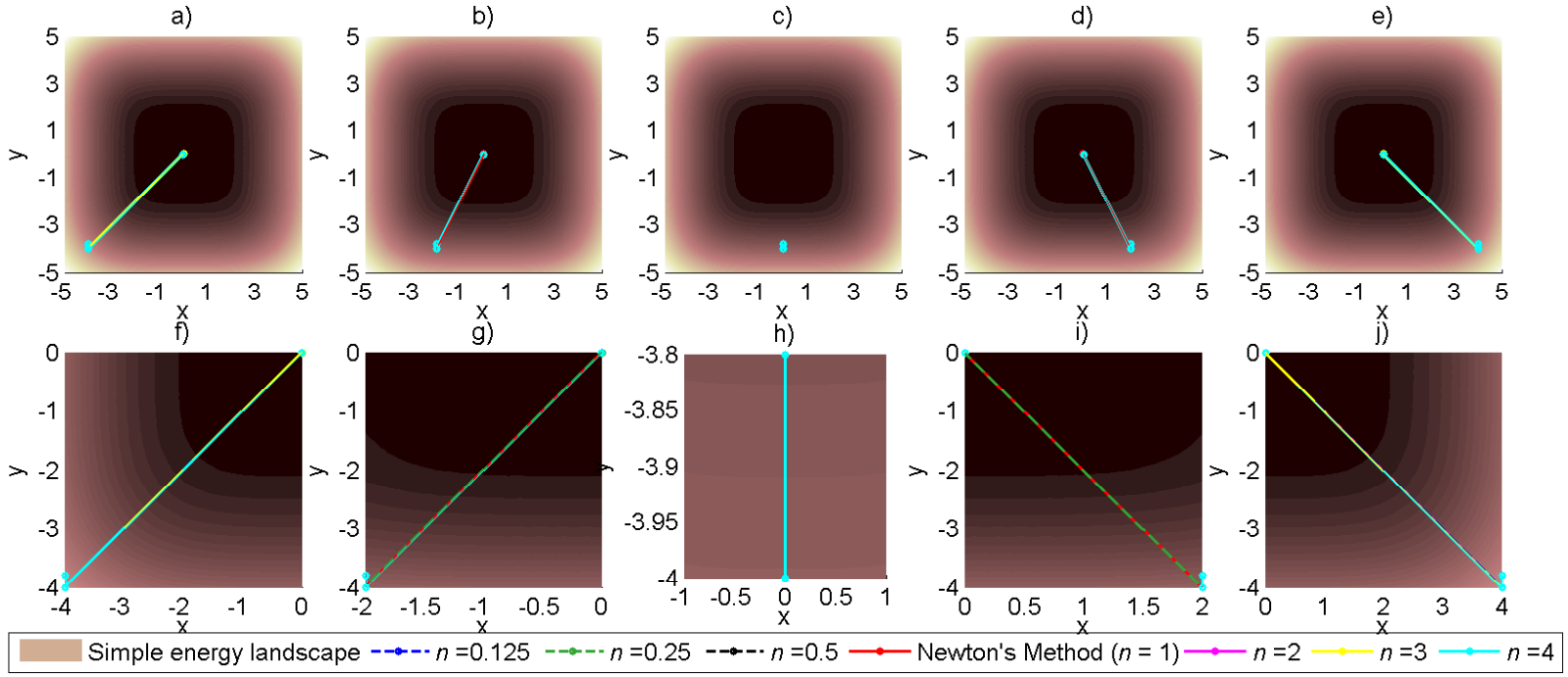

Figure 5.3: Paths from 1st through 5th starting points a) through e) on a simple energy landscape generated with Newton-XEL. f) through j) show more detail of the paths on a) through e), respectively. Paths are generated by a bracketing and Brent line search.

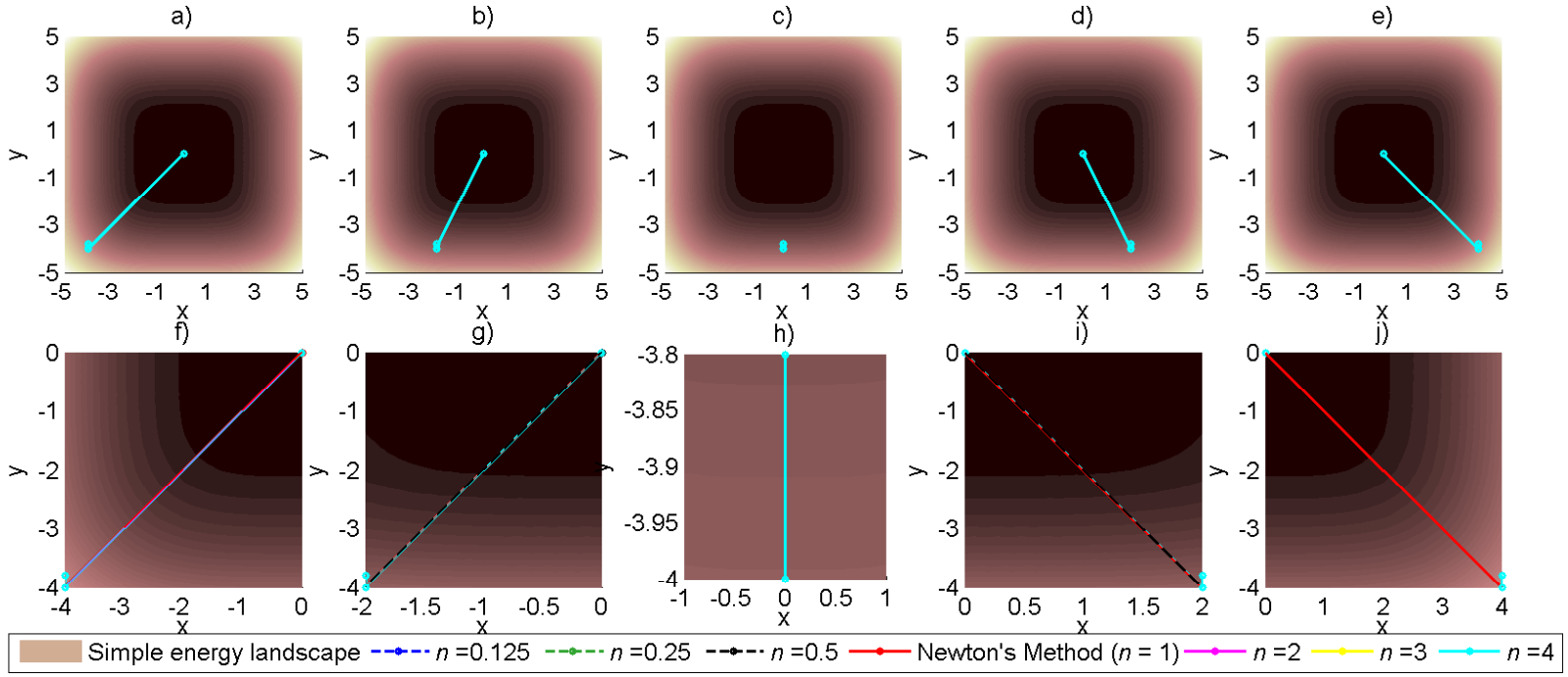

Figure 5.4: Paths from 1st through 5th starting points a) through e) on a simple energy landscape generated with Newton-XEL (*Hessian only*). f) through j) show more detail of the paths on a) through e), respectively. Paths are generated by a bracketing and Brent line search.

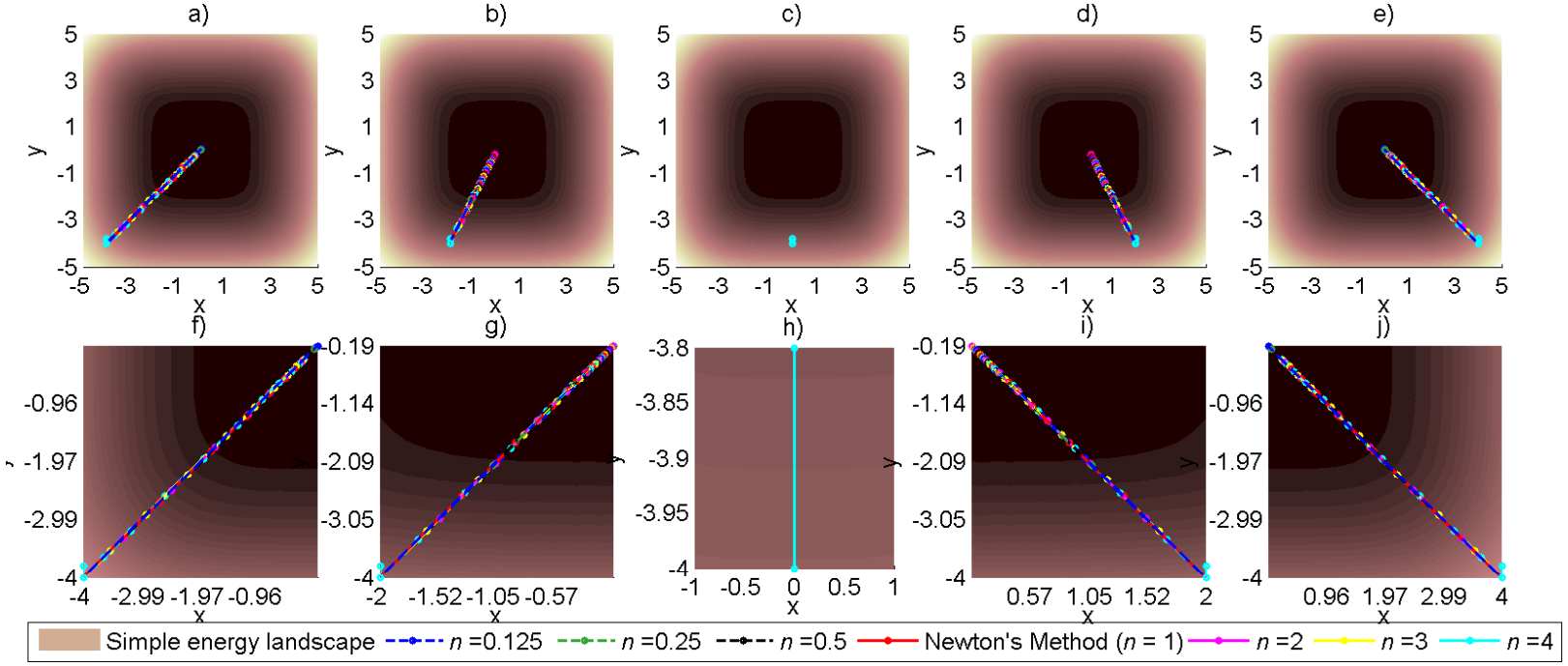

Figure 5.5: Paths from 1st through 5th starting points a) through e) on a simple energy landscape generated with Newton-XEL. f) through j) show more detail of the paths on a) through e), respectively. Paths are generated by a heuristic exhaustive line search.

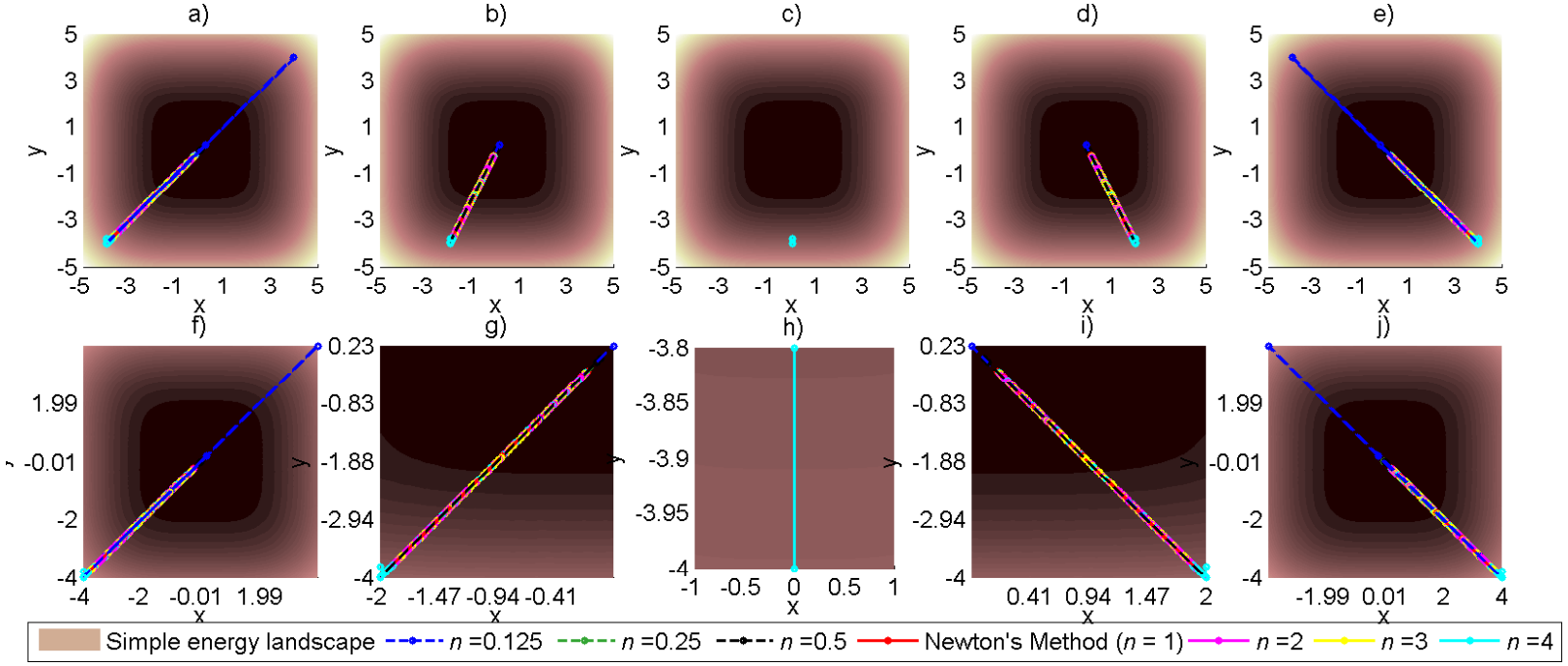

Figure 5.6: Paths from 1st through 5th starting points a) through e) on a simple energy landscape generated with Newton-XEL (*Hessian only*). f) through j) show more detail of the paths on a) through e), respectively. Paths are generated by a heuristic exhaustive line search.

Complex energy landscape

With the bracketing and Brent line search on the complex energy landscape, n has little effect on paths from Newton-XEL and Newton-XEL (*Hessian only*) as they rarely differ. This can be observed from Figures 5.7 and 5.8 with representative examples of paths from Newton-XEL and Newton-XEL (*Hessian only*) respectively. Paths from Newton-XEL (*Hessian only*) show more variation than those from Newton-XEL which demonstrates that n has more effect on Newton-XEL (*Hessian only*) than Newton-XEL. This is because the extra term β^{-1} in Newton-XEL (*Hessian only*) and with the exponent $n-1$ in $\beta = n\,|E|^{n-1}$ magnifies the dependency of n is magnified as discussed briefly in Section 4.1.4. Results here show that $n < 1$ yields a longer total step than $n > 1$ but the effects of n are not as pronounced as expected because the bracketing and Brent line search reduces the effect of n on the magnitudes of the step.

With the heuristic exhaustive line search, paths from different values of n on the complex energy landscape show more variation than those with the bracketing and Brent line search. This can be observed from Figures 5.9 and 5.10 with representative examples of paths from Newton-XEL and Newton-XEL (*Hessian only*) respectively. Again this is because the heuristic exhaustive line search does not significantly reduce the effect of n on the magnitude of the step. Paths from Newton-XEL (*Hessian only*) have larger total steps when $n < 1$ which allow paths to travel farther and more frequently reach the global minimum. However, with smaller total steps Newton-XEL paths tend to converge to the closest local minima.

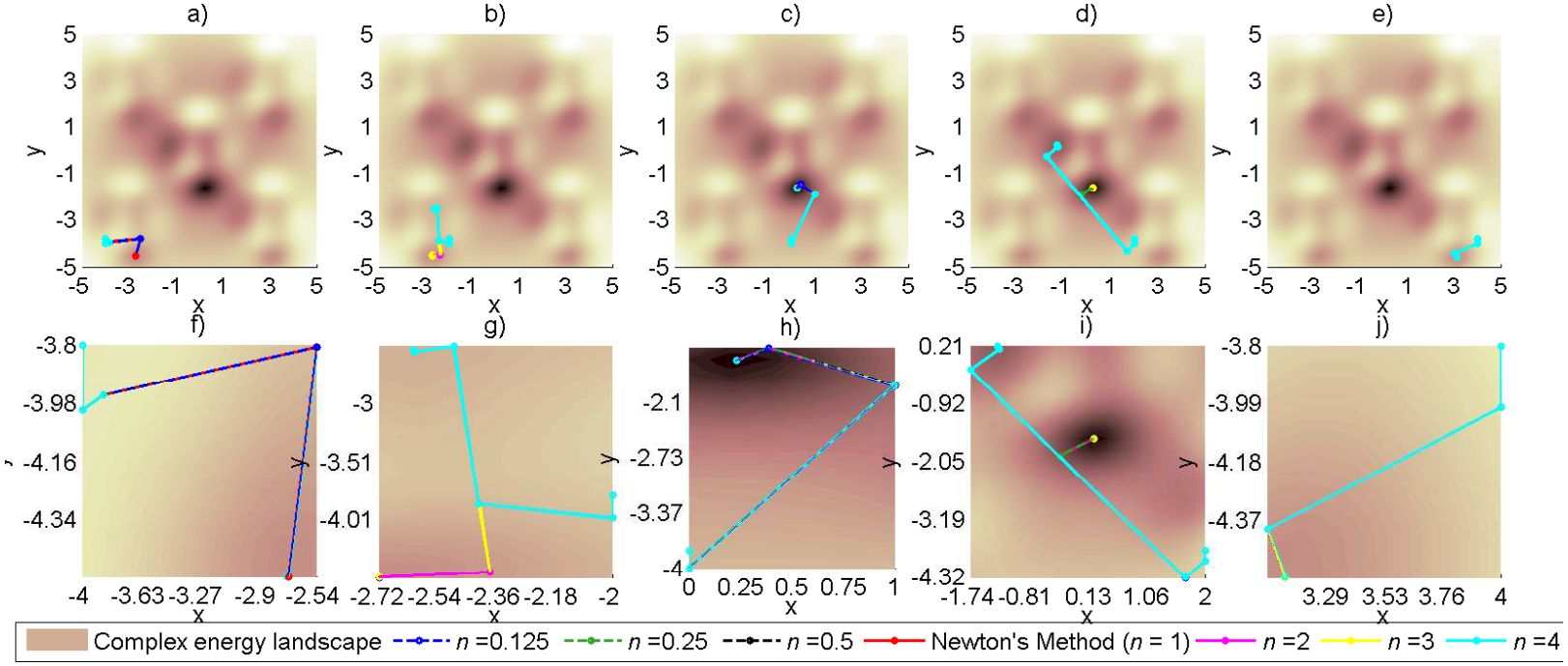

Figure 5.7: Paths from 1st through 5th starting points a) through e) on a complex energy landscape generated with Newton-XEL. f) through j) show more detail of the paths on a) through e), respectively. Paths are generated by a bracketing and Brent line search.

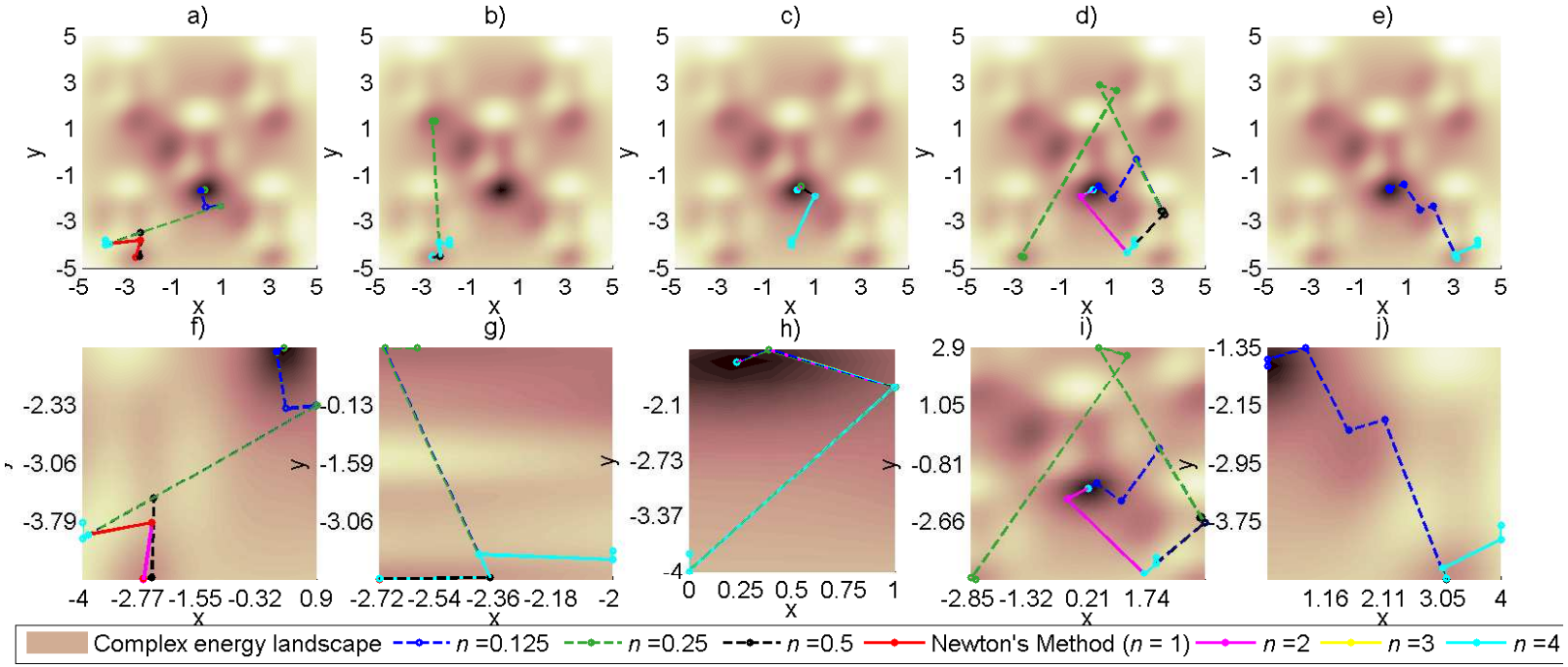

Figure 5.8: Paths from 1st through 5th starting points a) through e) on a complex energy landscape generated with Newton-XEL (*Hessian only*). f) through j) show more detail of the paths on a) through e), respectively. Paths are generated by a bracketing and Brent line search.

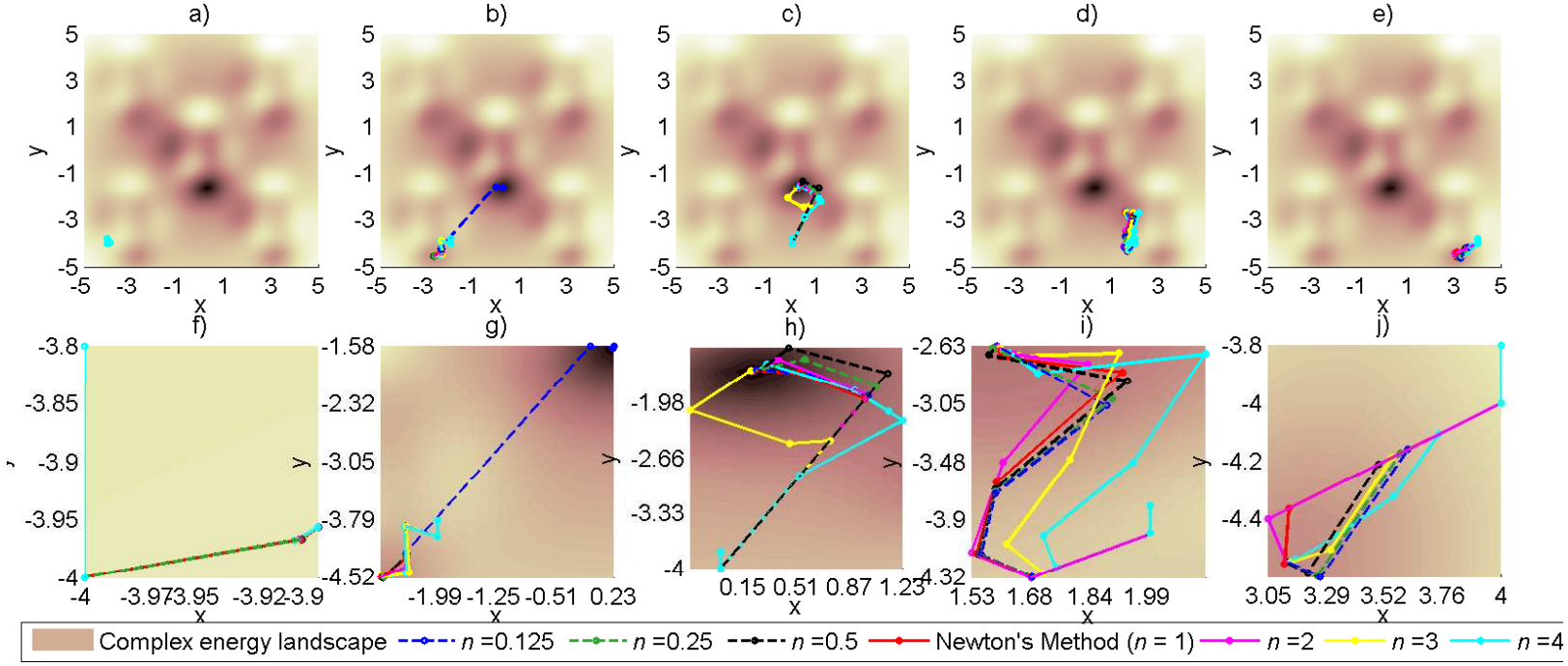

Figure 5.9: Paths from 1st through 5th starting points a) through e) on a complex energy landscape generated with Newton-XEL. f) through j) show more detail of the paths on a) through e), respectively. Paths are generated by a heuristic exhaustive line search.

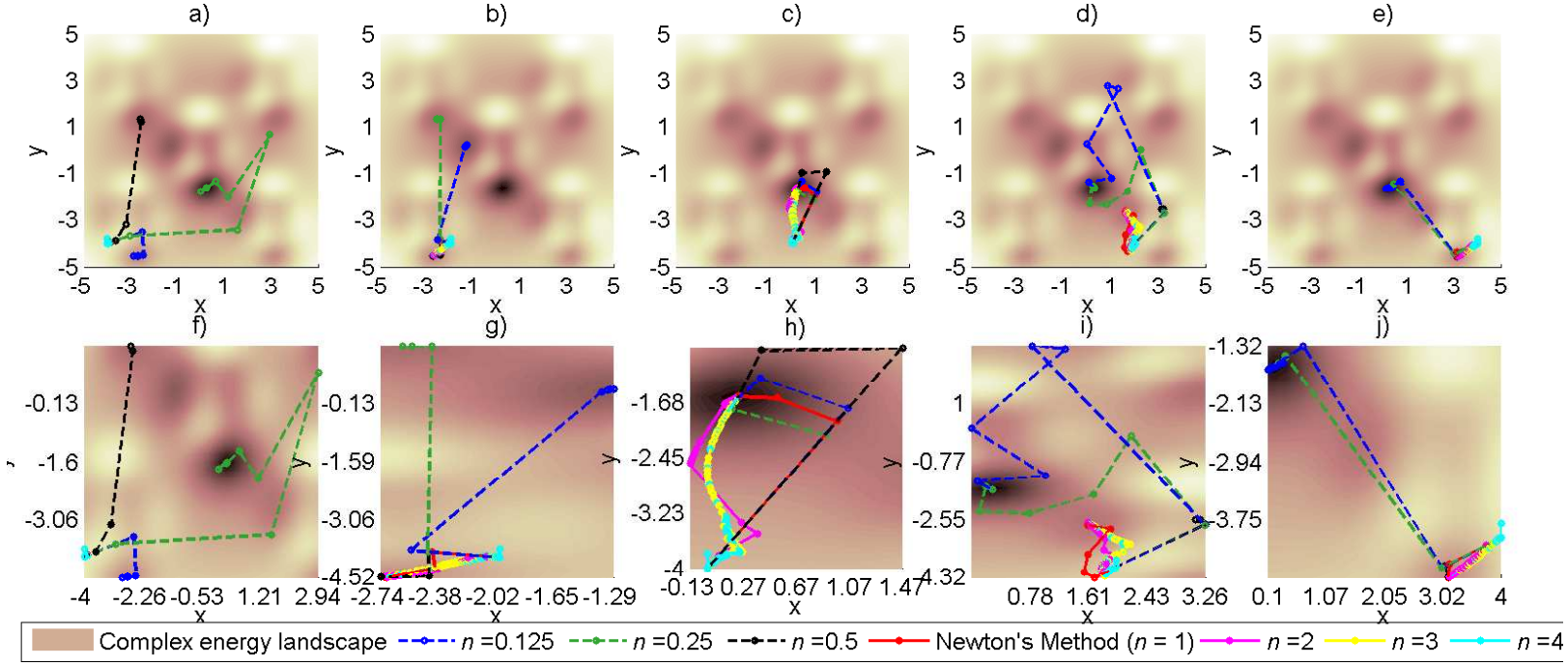

Figure 5.10: Paths from 1st through 5th starting points a) through e) on a complex energy landscape generated with Newton-XEL (*Hessian only*). f) through j) show more detail of the paths on a) through e), respectively. Paths are generated by a heuristic exhaustive line search.

5.1.1.2 QNA-XEL and QNA-XEL (Hessian only)

Simple energy landscape

With the bracketing and Brent line search on the simple energy landscape, QNA-XEL and QNA-XEL (*Hessian only*) (4.6) paths with different values of n differ significantly from each another. This can be observed from Figures 5.11 and 5.12 with representative examples of paths from QNA-XEL and QNA-XEL (*Hessian only*) respectively. Because both algorithms use Broyden-XEL (4.9), the step is significantly varied by the different values of n. However, the total steps are not varied by the different values of n because its effect is reduced by the bracketing and Brent line search.

Figures 5.11 and 5.12 also show that paths from QNA-XEL and QNA-XEL (*Hessian only*) are identical. Even though each algorithm gives a different magnitude of the step, the bracketing and Brent line search results in the same total step. The figures also show that many paths do not converge to the minimum. The paths stop before they reach the minimum because a new step can not be calculated when the line search gives a zero line search gain. This occurs when the step is almost perpendicular to the gradient which means the line search has a very small search interval so it converges with zero gain.

With a heuristic exhaustive line search, QNA-XEL and QNA-XEL (*Hessian only*) paths differ with n. This can be observed from Figures 5.13 and 5.14 with representative examples of paths from QNA-XEL and QNA-XEL (*Hessian only*) respectively. Paths from both algorithms show that when $n > 1$ the paths proceed directly toward the global minimum, similar to the Newton-XEL paths. This shows that the estimation of the Hessian matrix, $\hat{K}^*$, is very accurate. Since both the actual modified Hessian matrix, K^*, and the initial estimation of the Hessian matrix $\hat{K}_0^*$ for this energy function are diagonal, they are initially close. Because $n > 1$ paths have smaller total steps and the K^* is not varied too much along the path, it can be estimated more accurately.

Paths with a heuristic exhaustive line search also show that for $n \leq 1$ $\hat{K}^*$ becomes inaccurate due to the larger total step at the beginning of the path. As a result, paths are sometimes almost perpendicular to the negative gradient direction and occasionally diverge. As the QNA-XEL path becomes almost parallel to the negative gradient, the total step becomes very small because of the line search limiting the gain to a point with lowest energy. This means the change in gradient becomes very small and $\hat{K}^*$ is hardly updated. If $\hat{K}^*$ is close to the actual K^*, the small total step may not cause any problem. However, unlike at the beginning of the $n > 1$ paths, $\hat{K}^*$ here is far from the actual K^*, so it takes a long time for $\hat{K}^*$ to converge to K^* with too small a step size. These characteristics are not observed on paths with the bracketing and Brent line search because the magnitudes of their steps are not significantly varied by the different values of n. While most paths with the bracketing and Brent line search end at a close proximity of the minimum, they have many fewer total steps compared to paths with a heuristic exhaustive line search which implies higher speed.

While paths of the QNA-XEL (*Hessian only*) do not significantly vary from those of the QNA-XEL, Figure 5.14 shows that when $n = 0.125$ the step can diverge. This suggests that the QNA-XEL (*Hessian only*) may be less stable than the QNA-XEL for $n < 1$ as the total step can become too large.

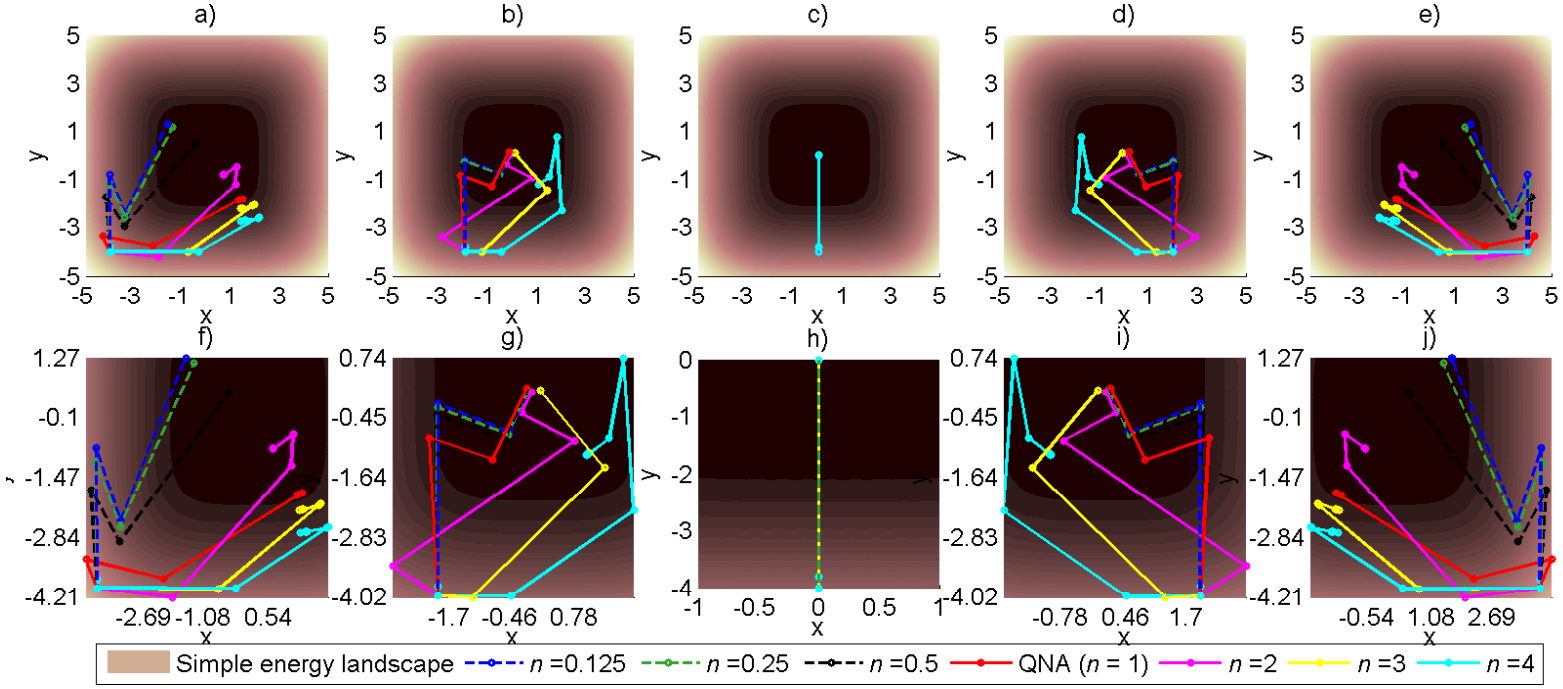

Figure 5.11: Paths from 1st through 5th starting points a) through e) on a simple energy landscape generated with QNA-XEL. f) through j) show more detail of the paths on a) through e), respectively. Paths are generated by a bracketing and Brent line search.

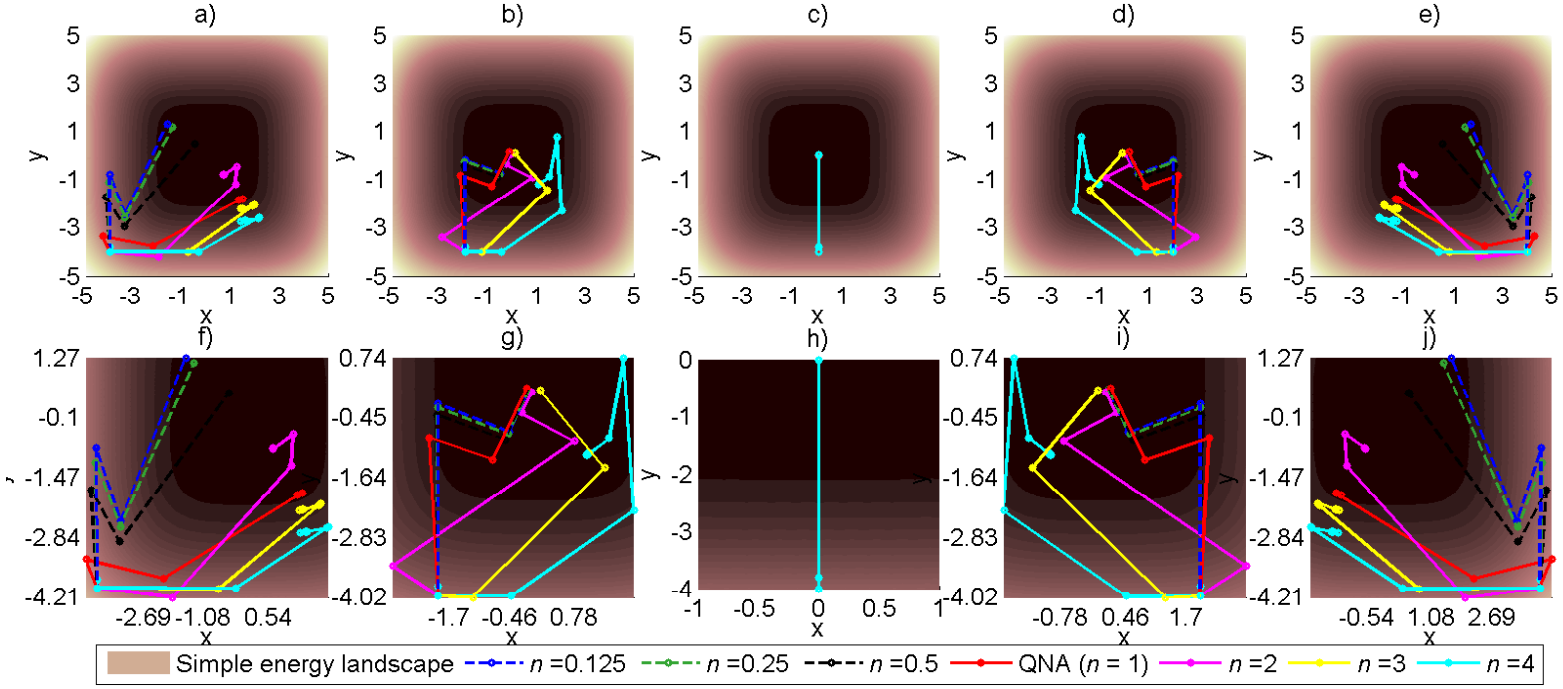

Figure 5.12: Paths from 1st through 5th starting points a) through e) on a simple energy landscape generated with QNA-XEL (*Hessian only*). f) through j) show more detail of the paths on a) through e), respectively. Paths are generated by a bracketing and Brent line search.

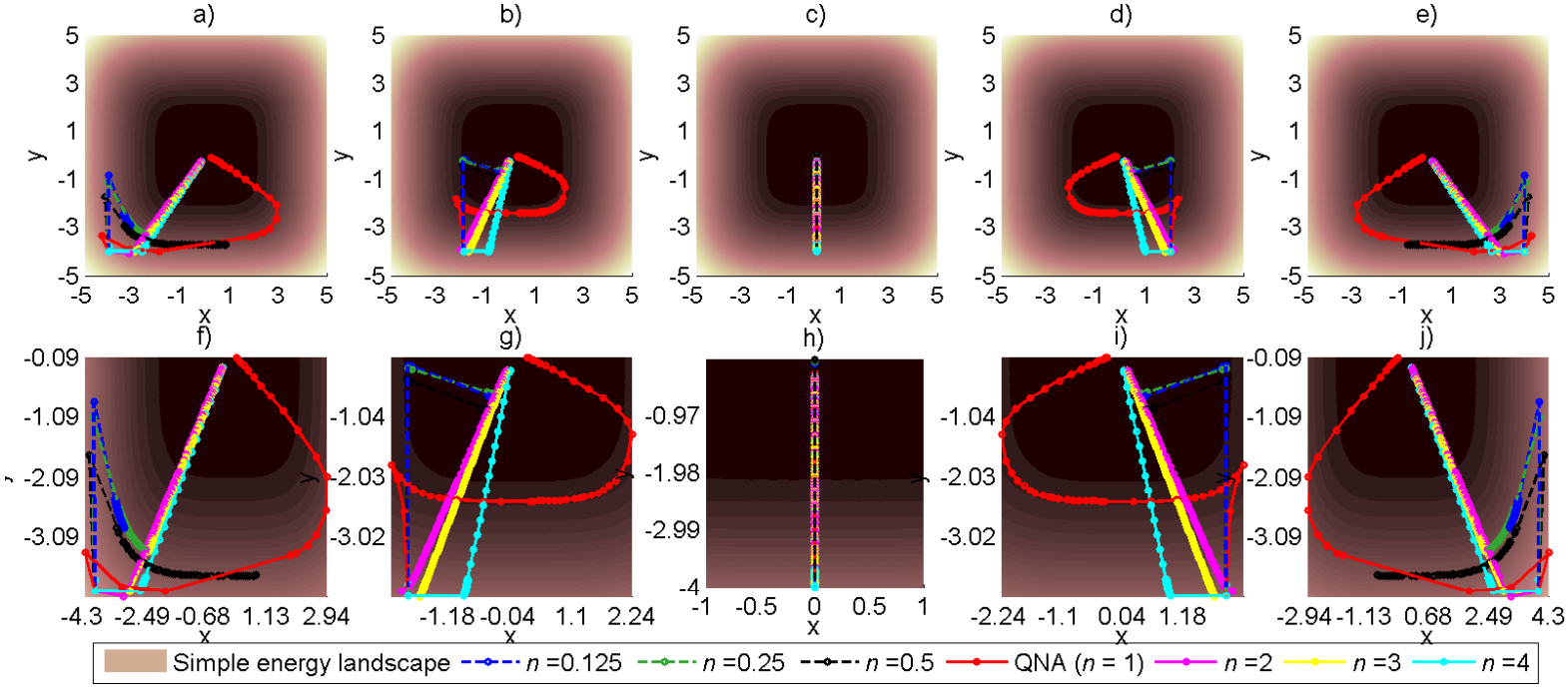

Figure 5.13: Paths from 1st through 5th starting points a) through e) on a simple energy landscape generated with QNA-XEL. f) through j) show more detail of the paths on a) through e), respectively. Paths are generated by a heuristic exhaustive line search.

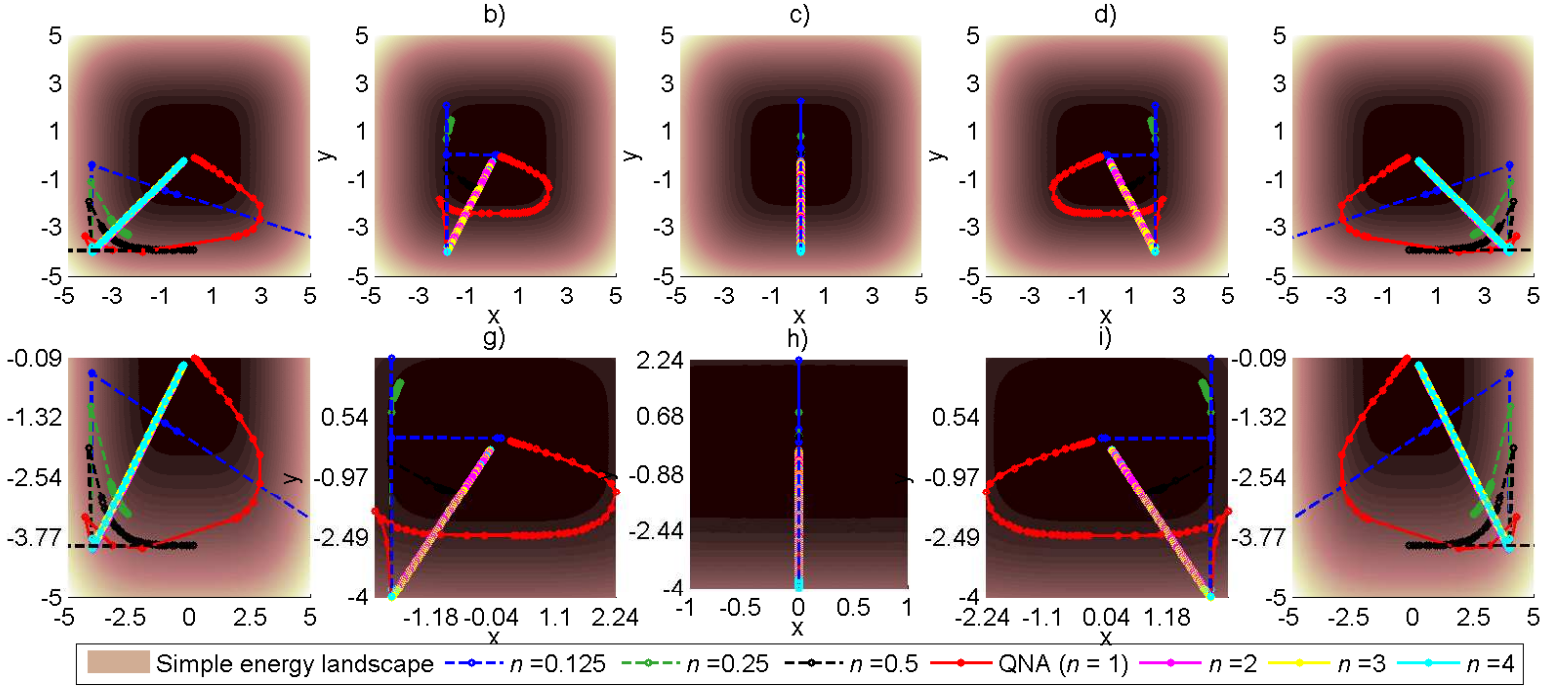

Figure 5.14: Paths from 1st through 5th starting points a) through e) on a simple energy landscape generated with QNA-XEL (*Hessian only*). f) through j) show more detail of the paths on a) through e), respectively. Paths are generated by a heuristic exhaustive line search.

Complex energy landscape

With the bracketing and Brent line search on the complex energy landscape, QNA-XEL and QNA-XEL (*Hessian only*) (4.6) paths with different values of n can differ significantly. This can be observed from Figures 5.15 and 5.16 with representative examples of paths from QNA-XEL and QNA-XEL (*Hessian only*) respectively. This is again because both algorithms use Broyden-XEL and the step is significantly varied by different values of n. Although QNA-XEL and QNA-XEL (*Hessian only*) give different magnitudes of the step which is varied by different values of n, the total steps and paths are similar because of the bracketing and Brent line search.

With a heuristic exhaustive line search, paths resulting from different values of n also differ from each other. This can be observed from Figures 5.17 and 5.18 with representative examples of paths from QNA-XEL and QNA-XEL (*Hessian only*) respectively. Similar to Newton-XEL paths on a complex energy landscape, QNA-XEL paths have smaller total steps for $n > 1$ and tend to converge to the closest local minima. On the contrary, paths have larger total steps for $n < 1$ and travel farther and more frequently reach the global minimum. However, when the total steps become too large paths are almost perpendicular to the gradient direction because of inaccurate $\hat{K}^*$. This case can be identified by the portions of paths that are perpendicular to the gradient and have small total steps which occurs more often as n gets smaller. For $n = 0.125$, paths sometimes diverge.

QNA-XEL and QNA-XEL (*Hessian only*) paths with a heuristic exhaustive line search show more deviation than those with the bracketing and Brent line search which implies that the heuristic exhaustive line search allows the magnitude of the step to have more effect on the paths. Paths with the heuristic exhaustive line search also have more total steps which implies that the heuristic exhaustive line search is less efficient than the bracketing and Brent line search.

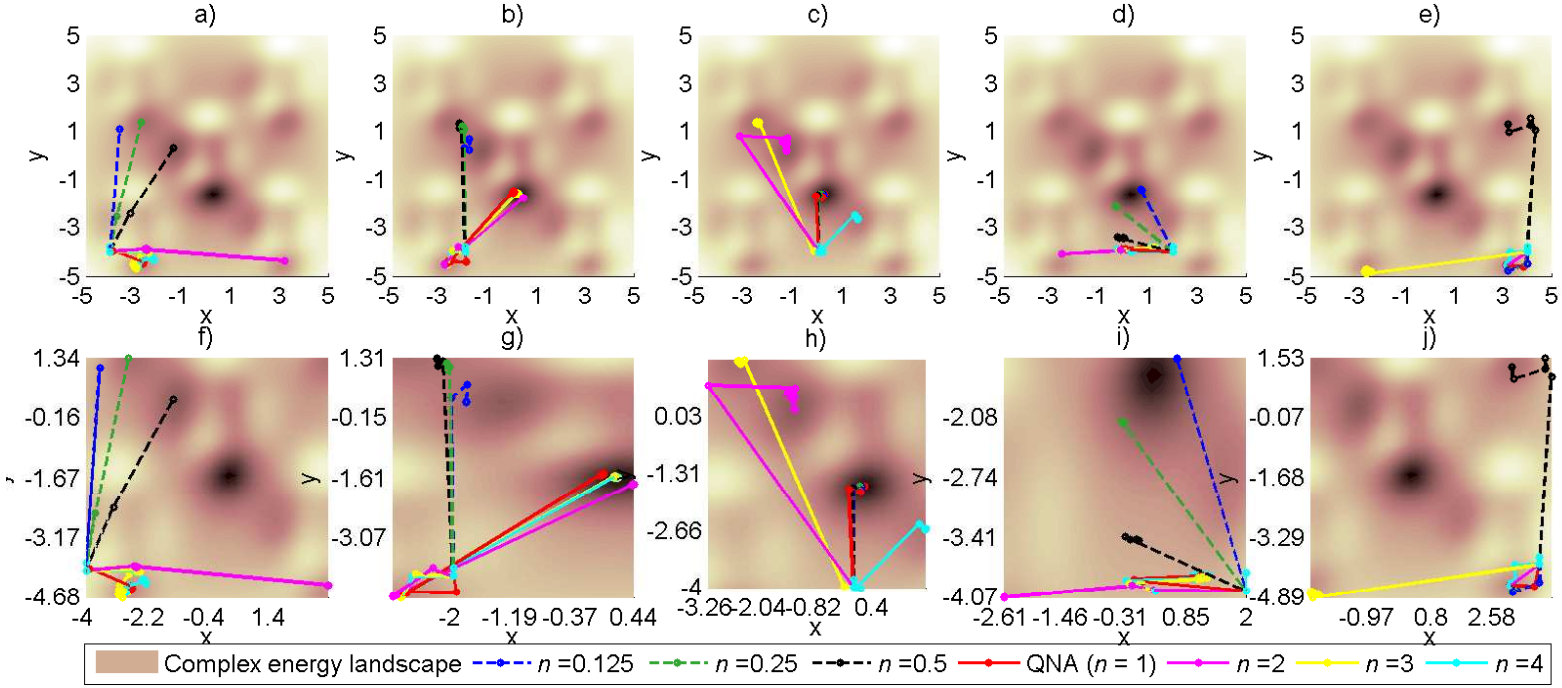

Figure 5.15: Paths from 1st through 5th starting points a) through e) on a complex energy landscape generated with QNA-XEL. f) through j) show more detail of the paths on a) through e), respectively. Paths are generated by a bracketing and Brent line search.

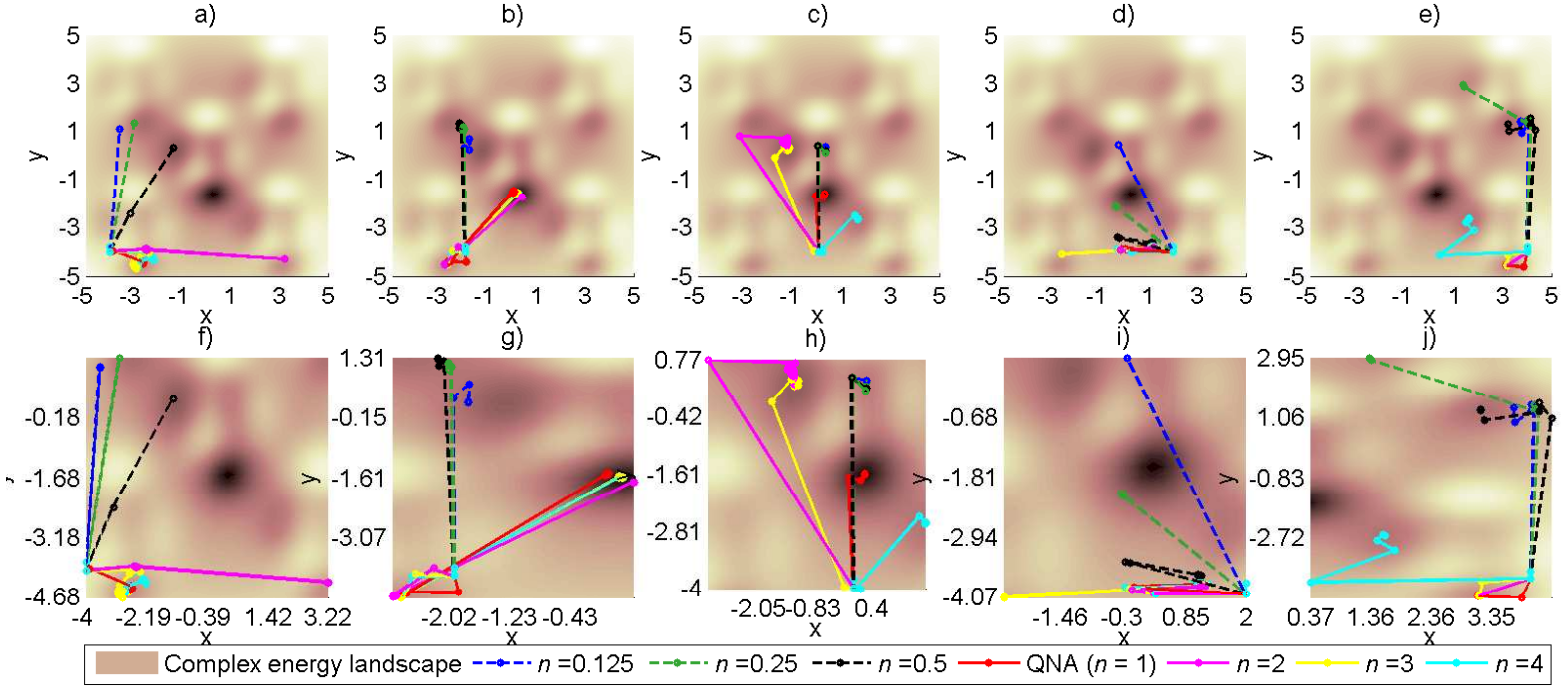

Figure 5.16: Paths from 1st through 5th starting points a) through e) on a complex energy landscape generated with QNA-XEL (*Hessian only*). f) through j) show more detail of the paths on a) through e), respectively. Paths are generated by a bracketing and Brent line search.

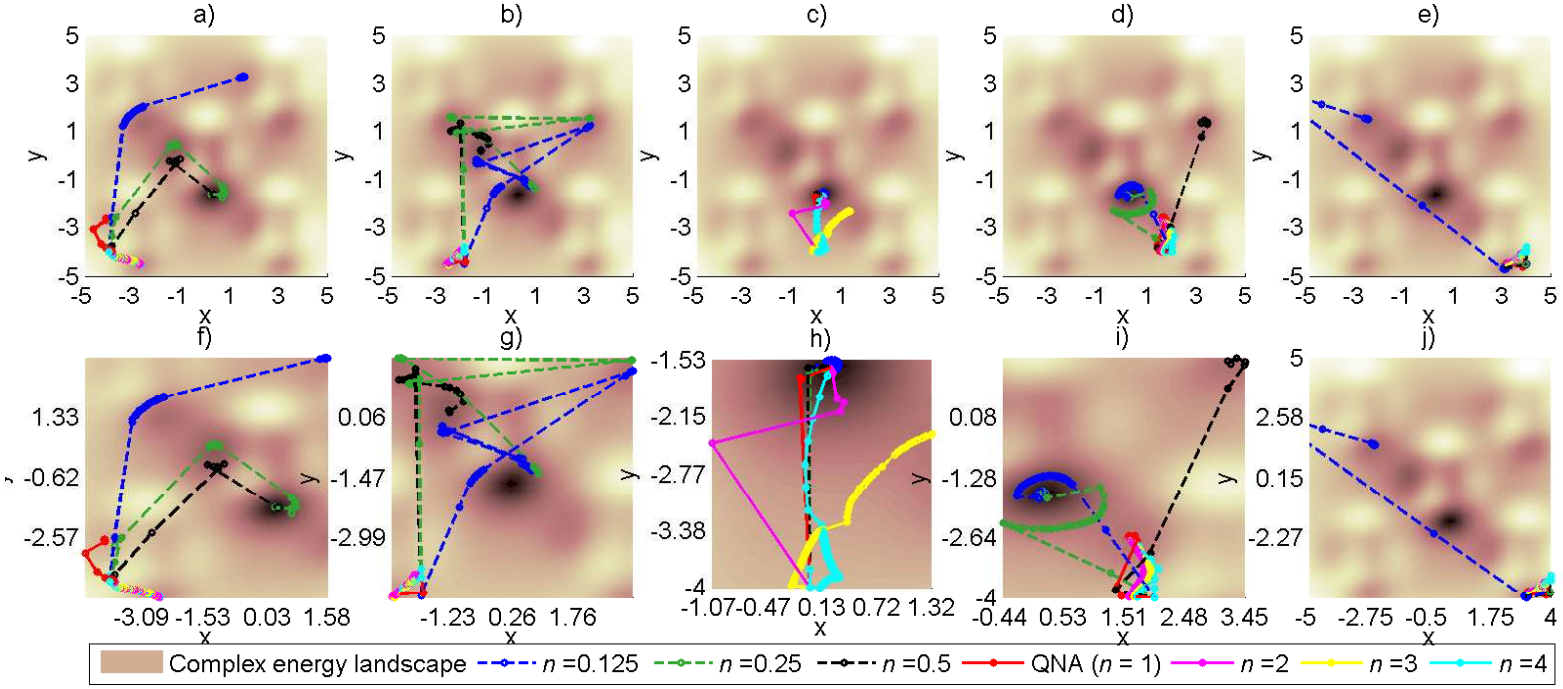

Figure 5.17: Paths from 1st through 5th starting points a) through e) on a complex energy landscape generated with QNA-XEL. f) through j) show more detail of the paths on a) through e), respectively. Paths are generated by a heuristic exhaustive line search.

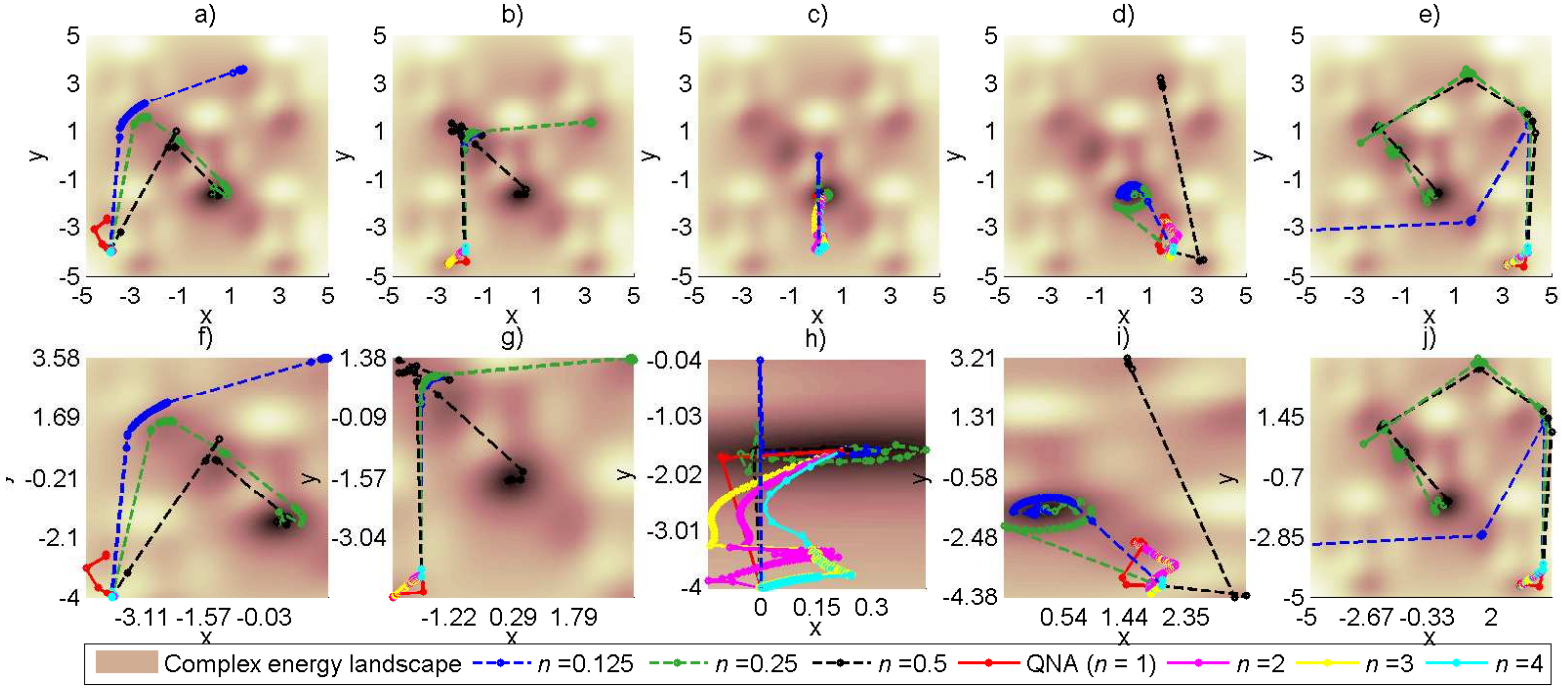

Figure 5.18: Paths from 1st through 5th starting points a) through e) on a complex energy landscape generated with QNA-XEL (*Hessian only*). f) through j) show more detail of the paths on a) through e), respectively. Paths are generated by a heuristic exhaustive line search.

5.1.1.3 BFGS-XEL and BFGS-XEL (*Hessian only*)

Simple energy landscape

With a bracketing and Brent line search on the simple energy landscape, BFGS-XEL and BFGS-XEL (*Hessian only*) (4.10) paths with different values of n can differ from one another. This can be observed from Figures 5.19 and 5.20 with representative examples of paths from BFGS-XEL and BFGS-XEL (*Hessian only*) respectively. The total step sizes of paths from both algorithms are not varied by the different values of n and their paths are identical due to the bracketing and Brent line search.

With a heuristic exhaustive line search, paths with different values of n on the simple energy landscape also differ from one another. This can be observed from Figures 5.21 and 5.22 with representative examples of paths from BFGS-XEL and BFGS-XEL (*Hessian only*) respectively. Unlike the Newton-XEL and the QNA-XEL paths on the simple energy landscape, after a few initial total steps in the BFGS-XEL paths the total step sizes are not varied with different values of n. After a few initial iterations, most $n \geq 1$ paths point straight toward the global minimum, but $n < 1$ paths do not. This indicates that $n \geq 1$ cases have more accurate estimations of the Hessian inverse, $\hat{H}^*$ than those of the $n < 1$ case which means that larger n can be used to improve the accuracy of $\hat{H}^*$.

With a heuristic exhaustive line search, paths from BFGS-XEL and BFGS-XEL (*Hessian only*) are significantly different. The step sizes of BFGS-XEL (*Hessian only*) paths are more influenced by n as the total steps are larger when $n < 1$ and less when $n > 1$. This difference is more pronounced than that between Newton-XEL and Newton-XEL (*Hessian only*) and between QNA-XEL and QNA-XEL (*Hessian only*). BFGS-XEL and BFGS-XEL (*Hessian only*) paths with a heuristic exhaustive line search also show more deviation than those with the bracketing and Brent line search which implies that the heuristic exhaustive line search allows the magnitude of the step to have more effect on the path. Paths with the heuristic exhaustive line search

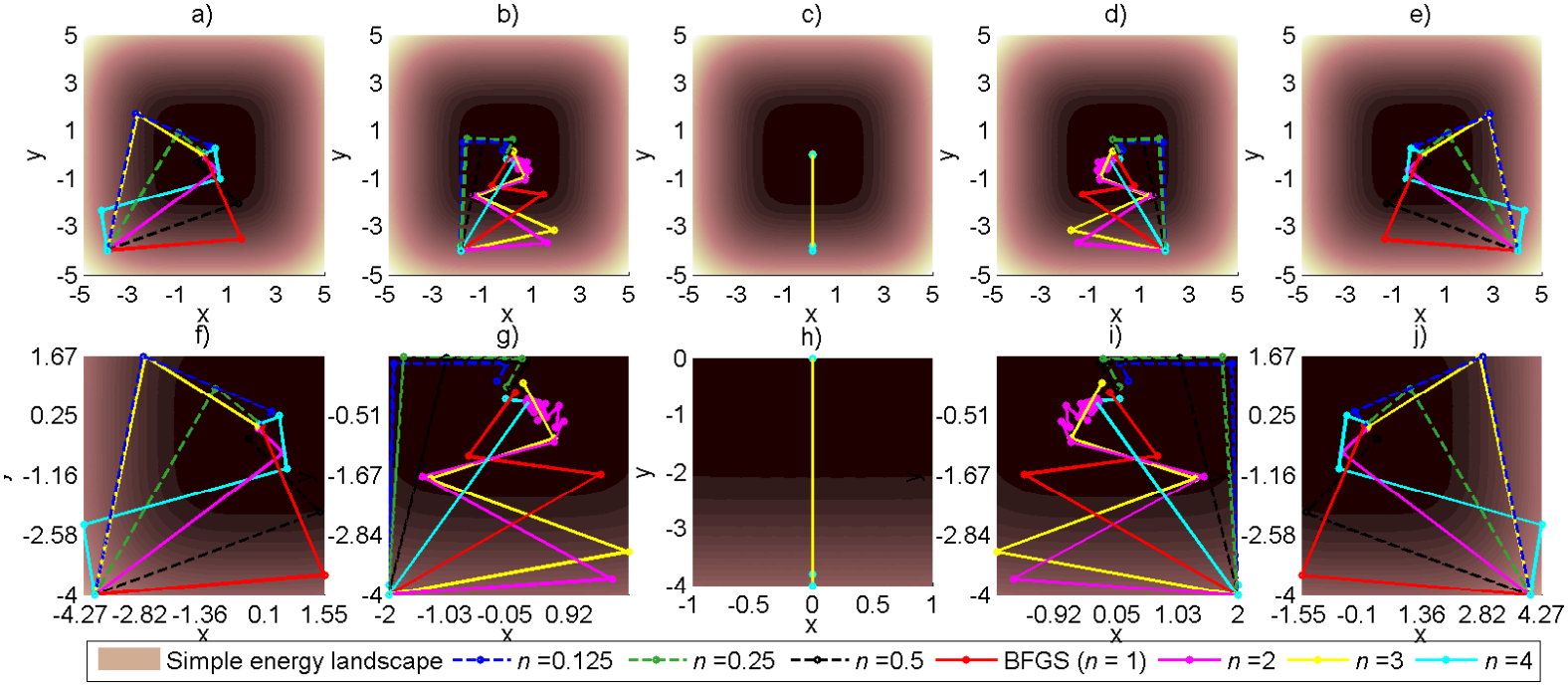

Figure 5.19: Paths from 1st through 5th starting points a) through e) on a simple energy landscape generated with BFGS-XEL. f) through j) show more detail of the paths on a) through e), respectively. Paths are generated by a bracketing and Brent line search.

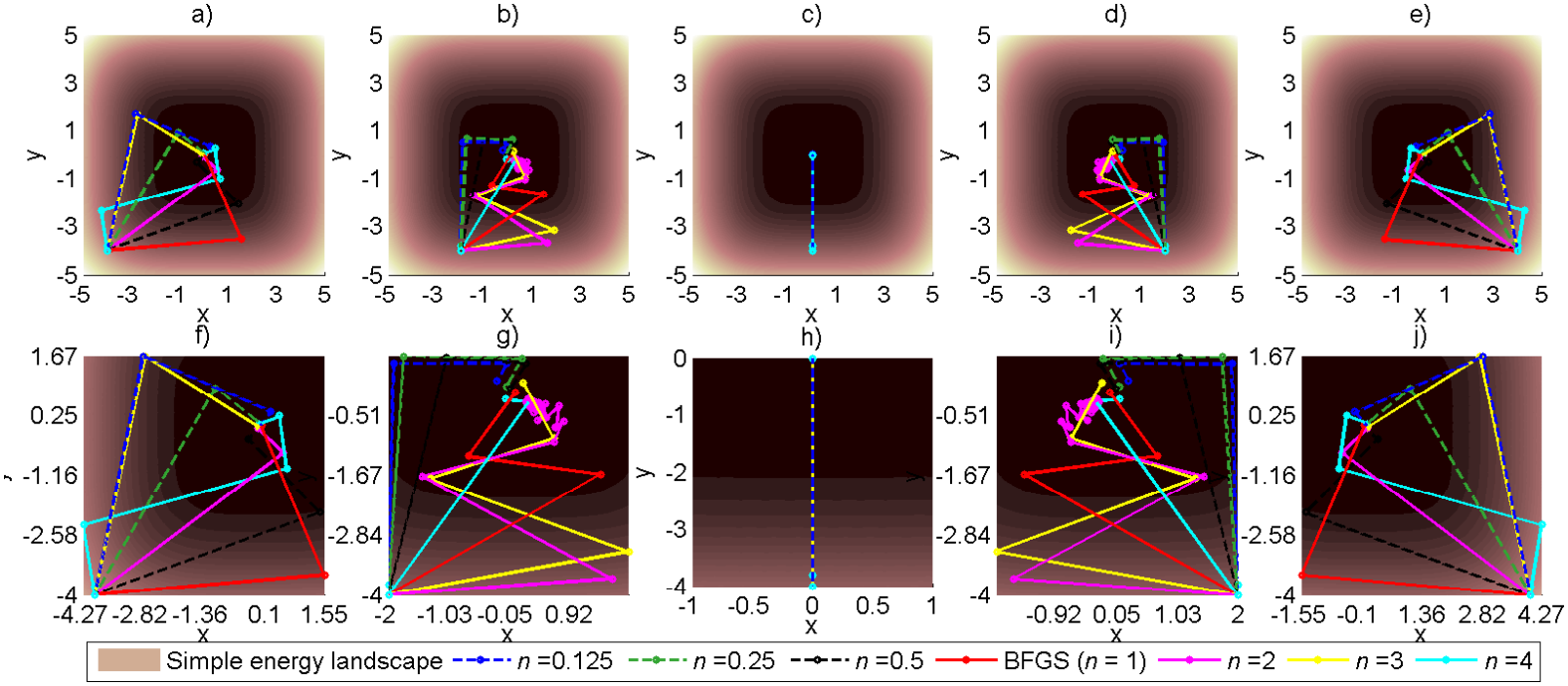

Figure 5.20: Paths from 1st through 5th starting points a) through e) on a simple energy landscape generated with BFGS-XEL (*Hessian only*). f) through j) show more detail of the paths on a) through e), respectively. Paths are generated by a bracketing and Brent line search.

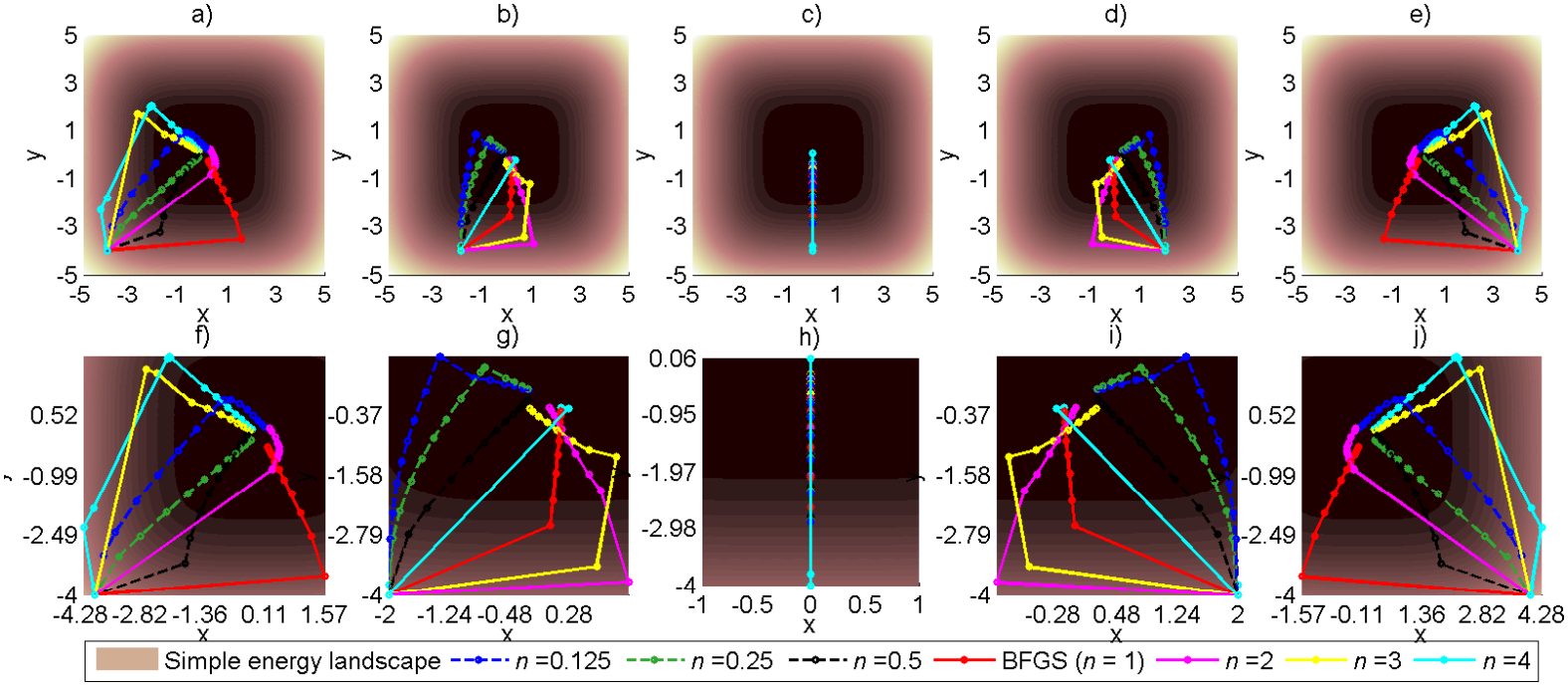

Figure 5.21: Paths from 1st through 5th starting points a) through e) on a simple energy landscape generated with BFGS-XEL. f) through j) show more detail of the paths on a) through e), respectively. Paths are generated by a heuristic exhaustive line search.

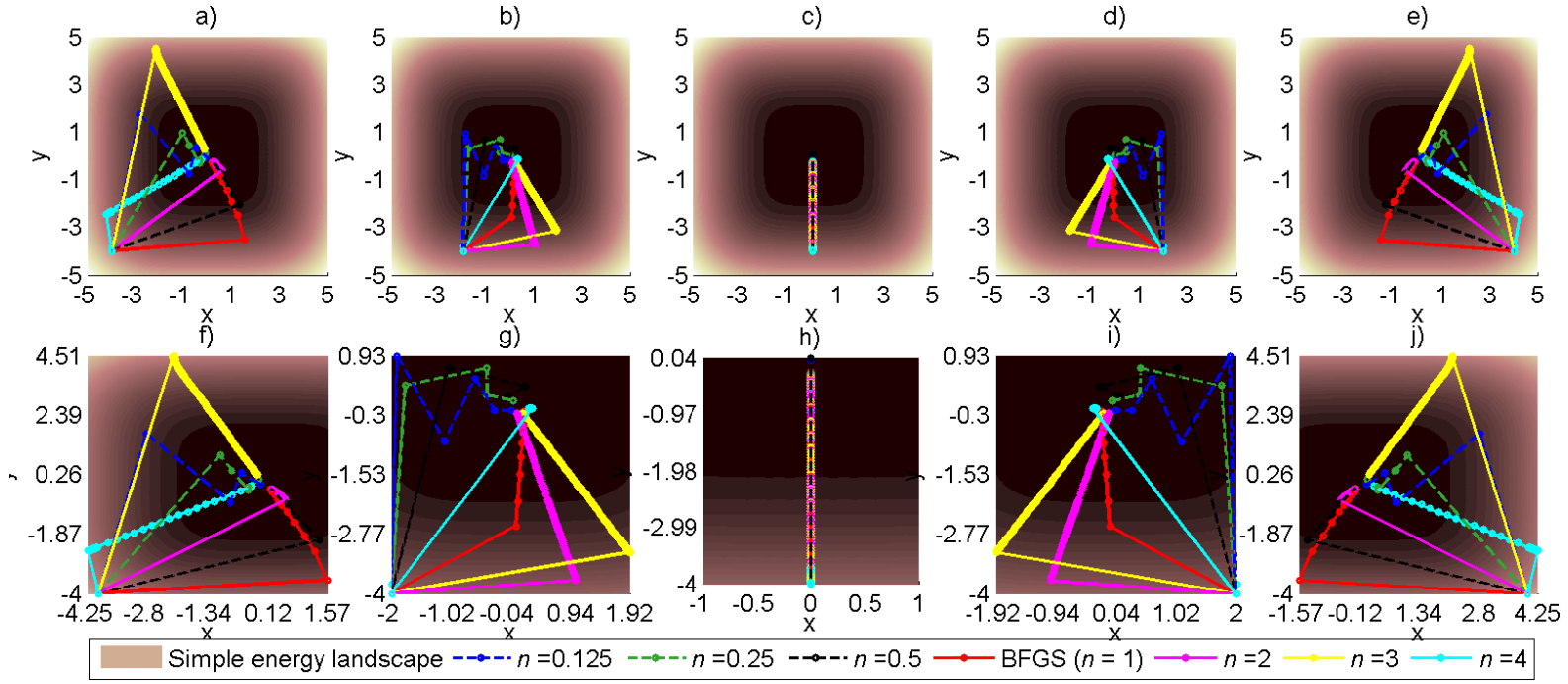

Figure 5.22: Paths from 1st through 5th starting points a) through e) on a simple energy landscape generated with BFGS-XEL (*Hessian only*). f) through j) show more detail of the paths on a) through e), respectively. Paths are generated by a heuristic exhaustive line search.

also have more steps which implies that the heuristic exhaustive line search is less efficient that the bracketing and Brent line search.

Complex energy landscape

With the bracketing and Brent line search on the complex energy landscape, BFGS-XEL and BFGS-XEL (*Hessian only*) paths with different values of n differ from one another. This can be observed from Figures 5.23 and 5.24 with representative examples of paths from BFGS-XEL and BFGS-XEL (*Hessian only*) respectively. While the step size appears to be in independent of n, the deviation between paths from BFGS-XEL and BFGS-XEL (*Hessian only*) is little noticeable.

With a heuristic exhaustive line search on the complex energy landscape, paths with different values of n differ from one another. This can be observed from Figures 5.25 and 5.26 with representative examples of paths from BFGS-XEL and BFGS-XEL (*Hessian only*) respectively. Unlike Newton-XEL and the QNA-XEL paths on a complex energy landscape, the relationship between the step size and n on the BFGS-XEL paths is less distinguishable.

With a heuristic exhaustive line search, BFGS-XEL and BFGS-XEL (*Hessian only*) paths also show slightly more deviation than those with the bracketing and Brent line search which implies that the heuristic exhaustive line search allows the magnitude of the step to have more effect on paths. Paths from both line search algorithms appear to have about the same number of steps which implies that both line search algorithms have about the same effect on the the BFGS-XEL and BFGS-XEL (*Hessian only*) in terms of speed.

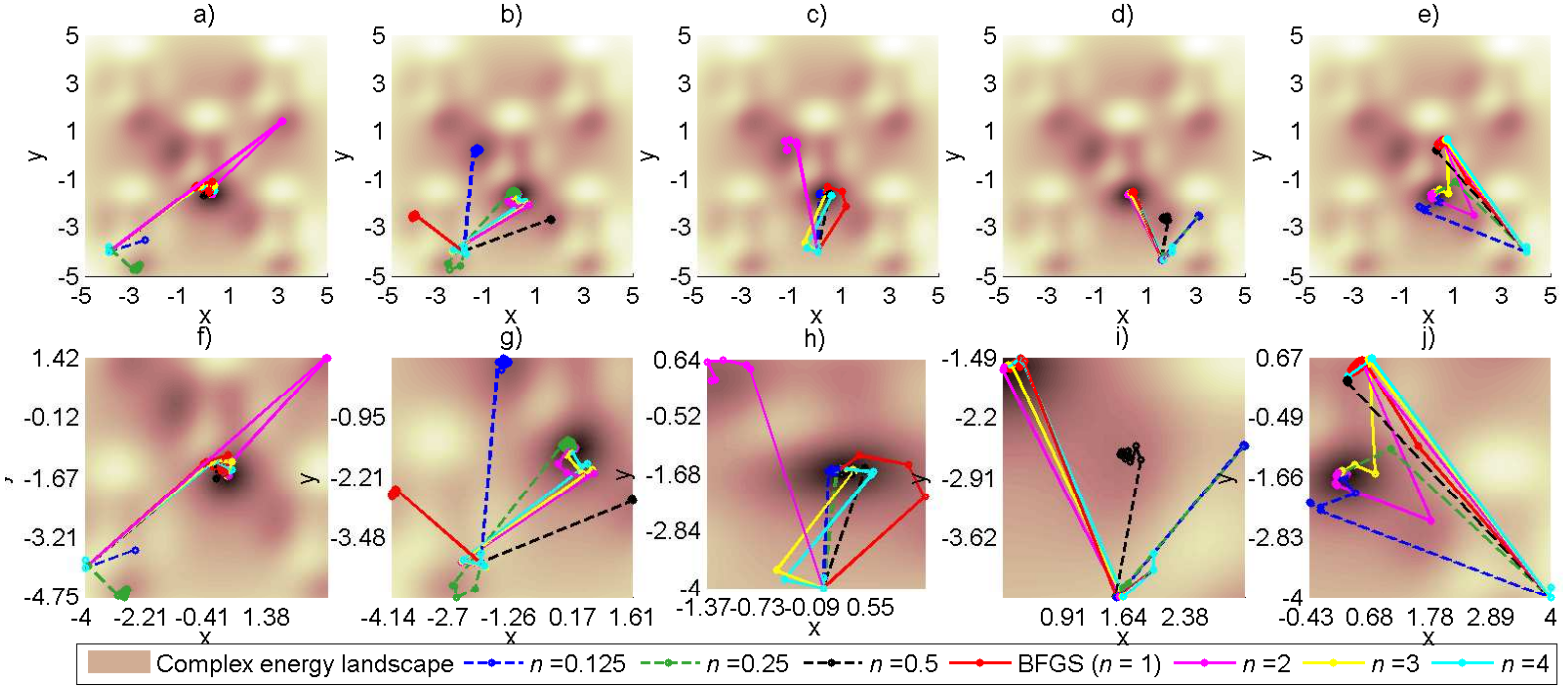

Figure 5.23: Paths from 1st through 5th starting points a) through e) on a complex energy landscape generated with BFGS-XEL. f) through j) show more detail of the paths on a) through e), respectively. Paths are generated by a bracketing and Brent line search.

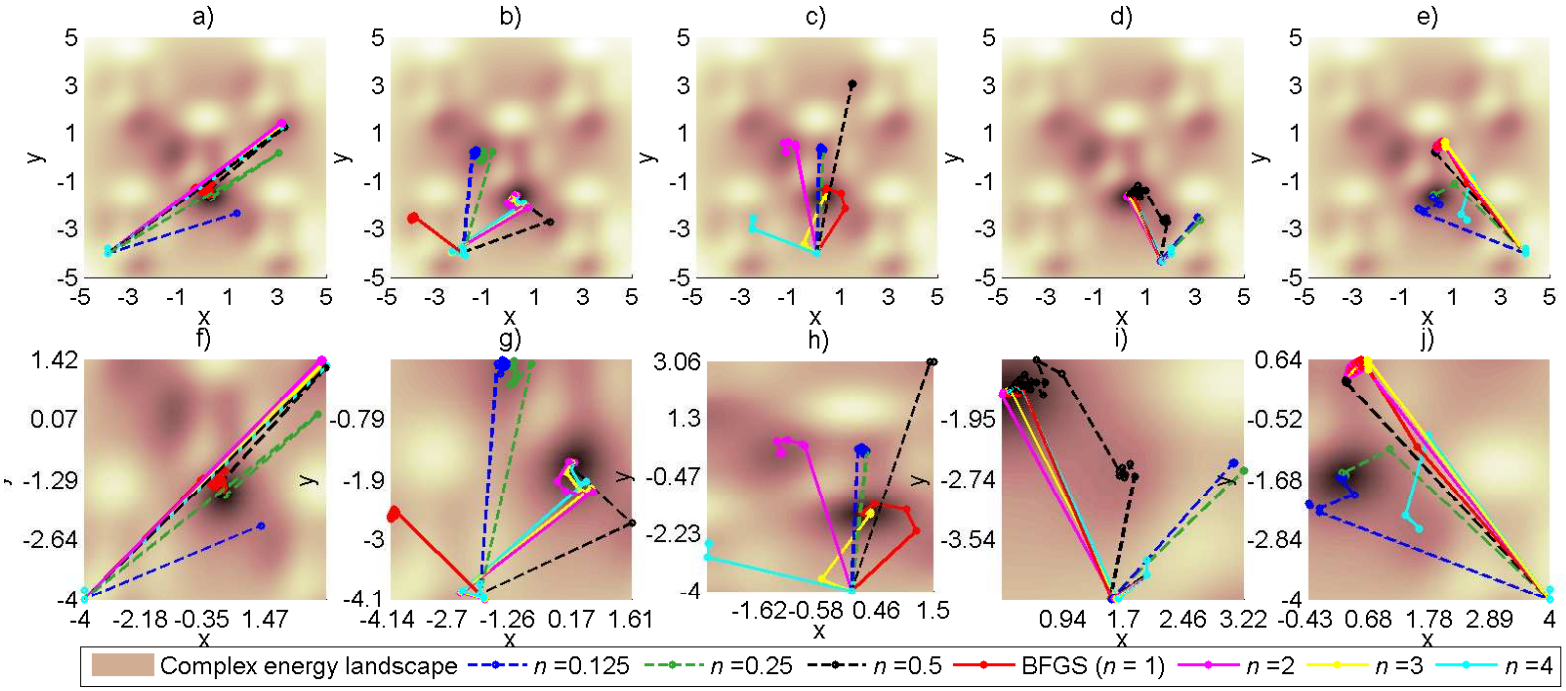

Figure 5.24: Paths from 1st through 5th starting points a) through e) on a complex energy landscape generated with BFGS-XEL (*Hessian only*). f) through j) show more detail of the paths on a) through e), respectively. Paths are generated by a bracketing and Brent line search.

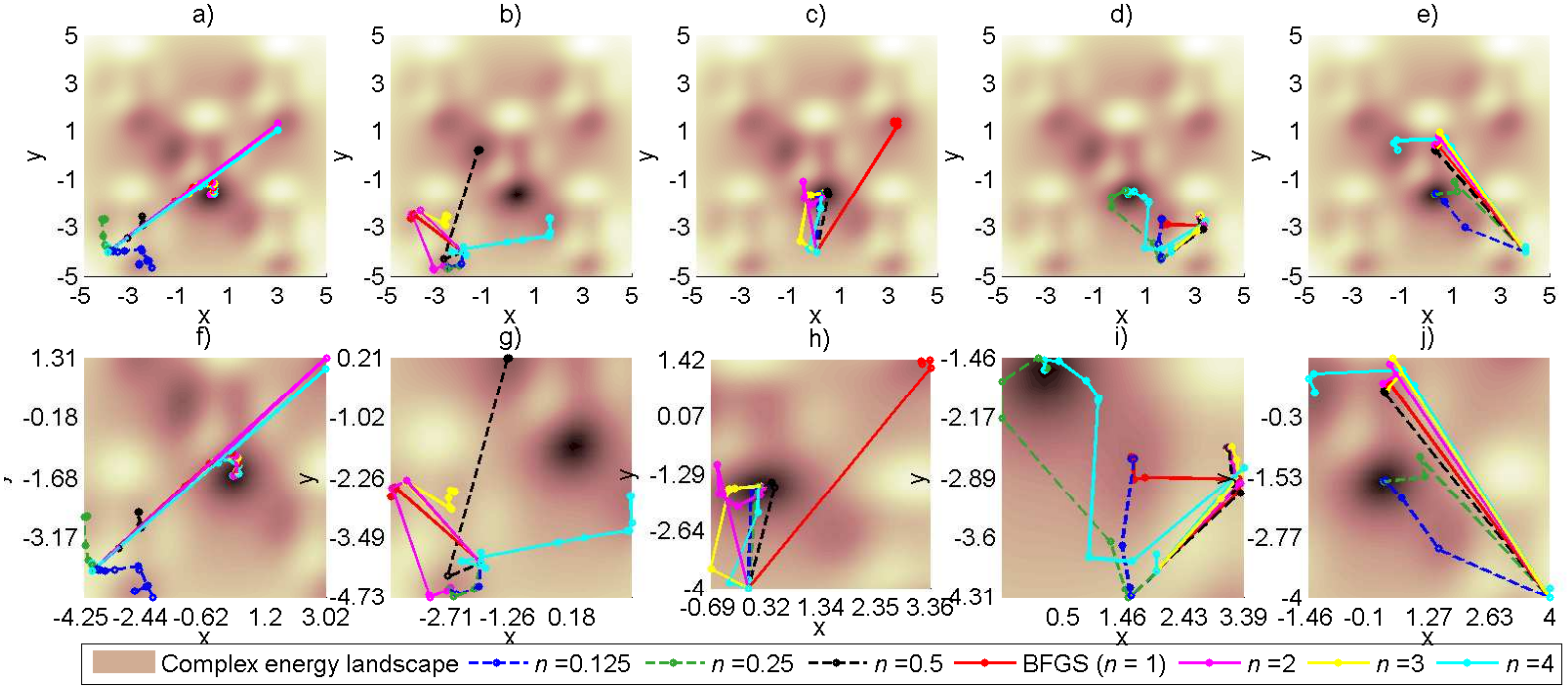

Figure 5.25: Paths from 1st through 5th starting points a) through e) on a complex energy landscape generated with BFGS-XEL. f) through j) show more detail of the paths on a) through e), respectively. Paths are generated by a heuristic exhaustive line search.

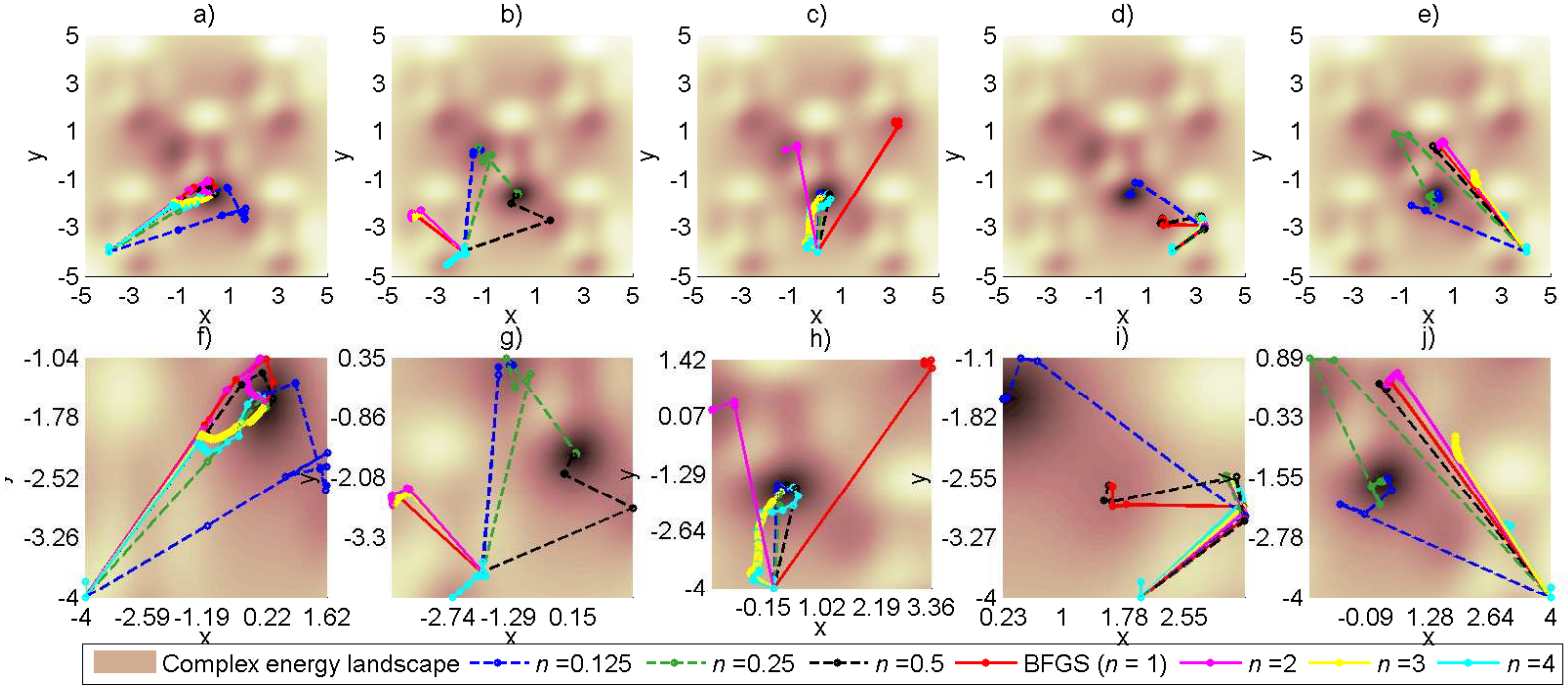

Figure 5.26: Paths from 1st through 5th starting points a) through e) on a complex energy landscape generated with BFGS-XEL (*Hessian only*). f) through j) show more detail of the paths on a) through e), respectively. Paths are generated by a heuristic exhaustive line search.

5.1.1.4 Conclusion of path comparison

The characteristics of paths on the simple and the complex energy landscape from the Newton-XEL, Newton-XEL (*Hessian only*), QNA-XEL, QNA-XEL (*Hessian only*), BFGS-XEL, and BFGS-XEL (*Hessian only*) with a bracketing and Brent line search and a heuristic exhaustive line search are compared and summarized on Table 5.1. Results show that paths are dependent on n with an exception of Newton-XEL paths on a simple energy landscape. The step size can show some dependency on n but it can be reduced by the bracketing and Brent line search and it is less apparent on the complex landscape. The XEL (*Hessian only*) algorithms yield different paths from the XEL algorithms because they give different magnitudes of the step but the effect can be reduced by the bracketing and Brent line search. The number of steps from paths with the bracketing and Brent line search are less than those from paths with the heuristic exhaustive line search which means the bracketing and Brent line search is more efficient. Lastly, the QNA-XEL and the BFGS-XEL paths suggest that $n > 1$ results in more accurate Hessian matrix estimation which is in agreement with Guideline *iv.* in Table 4.1.

To compare the quality and speed of different algorithms, the next section compares energy and number of iterations as the percentage of paths with lower, equal, or higher values compared to $n = 1$.

5.1.2 Comparison of Energy and Number of Iterations to the Unmodified Case

For each simulation, the converged energy and the number of iterations (or steps) are evaluated and compared to those of the unmodified case ($n = 1$) to assess quality and speed. The percentage of paths from each XEL algorithm with different n cases that has lower, equal, or higher energy or number of iterations than those of $n = 1$ on each energy landscape are determined. The values are considered "equal" if they are within $\pm 5\%$ of those of $n = 1$, "lower" if they are lower than the range, and "higher"

Table 5.1: A summary of the path characteristics from implemented algorithms.

	Characteristic	Newton-XEL		QNA-XEL		BFGS-XEL	
		Brent	**Exhaustive**	**Brent**	**Exhaustive**	**Brent**	**Exhaustive**
Simple Energy Landscape	Paths are varied by different values of n	No	No	Yes	Yes	Yes	Yes
	Step size are varied by different values of n	No	Yes	No	Yes	No	Some
	Number of steps compared to that from the other line search	Less	More	Less	More	Less	More
	Paths differ from (*Hessian only*)	No	Few	No	Few	No	Yes
Complex Energy Landscape	Paths are varied by different values of n	Few	Most	Yes	Yes	Yes	Yes
	Step size are varied by different values of n	No	No	No	Yes	No	No
	Number of steps compared to that from the other line search	Less	More	Less	More	Same	Same
	Paths differ from (*Hessian only*)	Some	Most	Some	Most	Some	Most

if they are above the range. Results are shown in Figures 5.27 through 5.38.

5.1.2.1 Newton-XEL and Newton-XEL (Hessian only)

Figures 5.27 through 5.30 display the percentage of paths from Newton-XEL and Newton-XEL (*Hessian only*) with different n cases that have smaller, equal, or higher number of iterations or energy than those of $n = 1$ on the simple and the complex energy landscape. While paths resulting in Figures 5.27 and 5.28 are generated by a bracketing and Brent line search, paths resulting in Figures 5.29 and 5.30 are generated by a heuristic exhaustive line search. The higher percentage of paths with smaller energy or number of iterations implies better quality and speed.

With a bracketing and Brent line search, the Newton-XEL (Figure 5.27 a and c) on the simple energy landscape have the same quality and speed for all n as all paths have equal energy and number of iterations compared to $n = 1$. On the complex energy landscape (Figure 5.27 b and d) quality and speed become slightly worse. Although up to 20% of paths for all n cases from the Newton-XEL have lower energy than $n = 1$, 12% to 40% have higher energy and up to 30% have higher number of iterations.

With a bracketing and Brent line search, the Newton-XEL (*Hessian only*) (Figure 5.28 a and c) on the simple energy landscape also have the same quality and speed for all n as all paths have equal energy and number of iterations compared to $n = 1$. On the complex energy landscape (Figure 5.28 b and d) quality becomes slightly better in some n cases and speed becomes slightly worse. When $n < 1$, 8% to 16% of paths have higher energy than $n = 1$ but 24% to 36% have lower energy, and when $n > 1$, 24% to 30% have higher energy and 12% to 16% have lower energy. For all n cases 12% to 32% of paths have a higher number of iterations while only 4% to 12% have lower number of iterations. Results from Newton-XEL (*Hessian only*) show slightly more deviation for different values of n because n has more effect on the magnitude

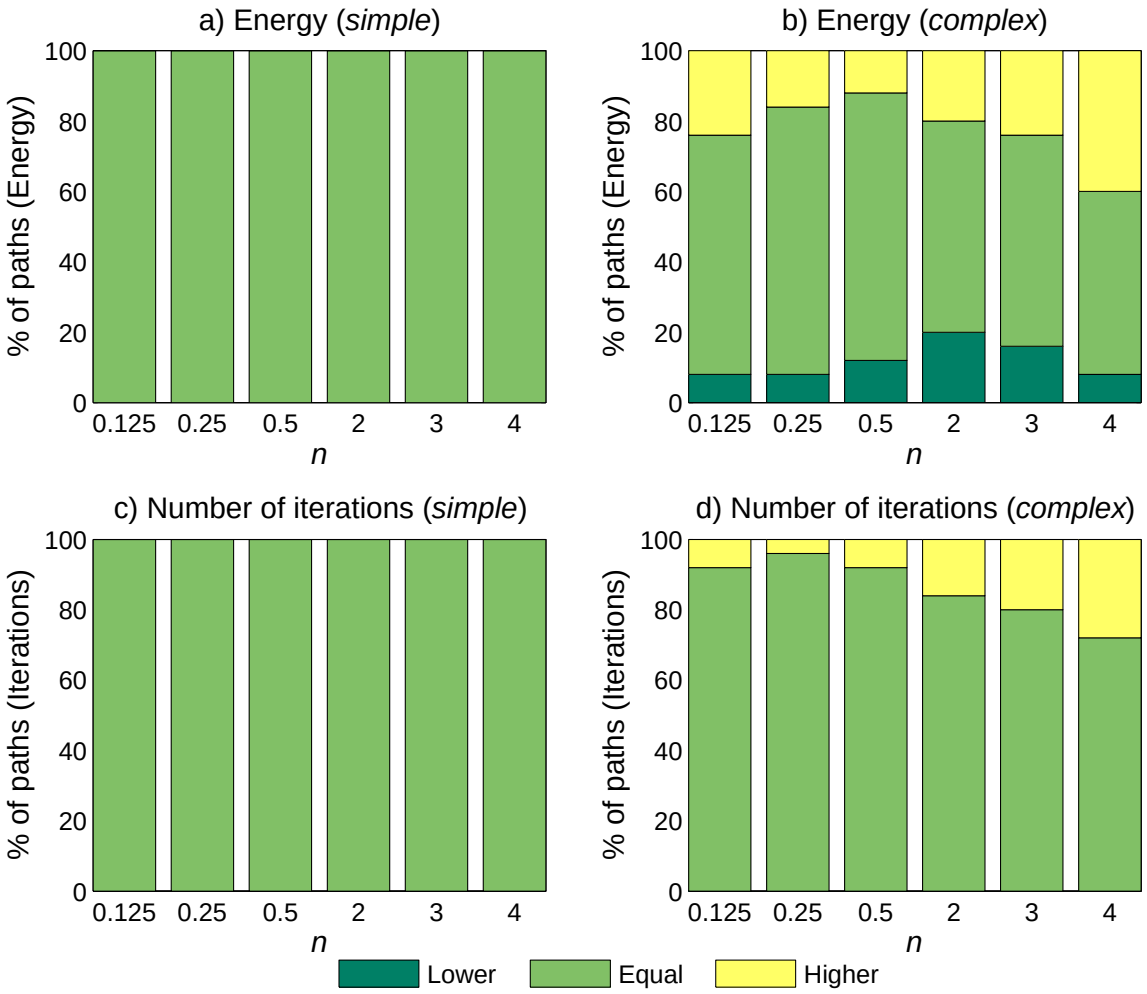

Figure 5.27: Newton-XEL: The percentage of paths that yields the lower, the equal, or the higher number of iterations in c) and d) or energy in a) and b) compared to $n = 1$. Shown in a) and c) are the percentage of paths on the simple energy landscape and in b) and d), on the complex energy landscape. Paths are generated by a bracketing and Brent line search.

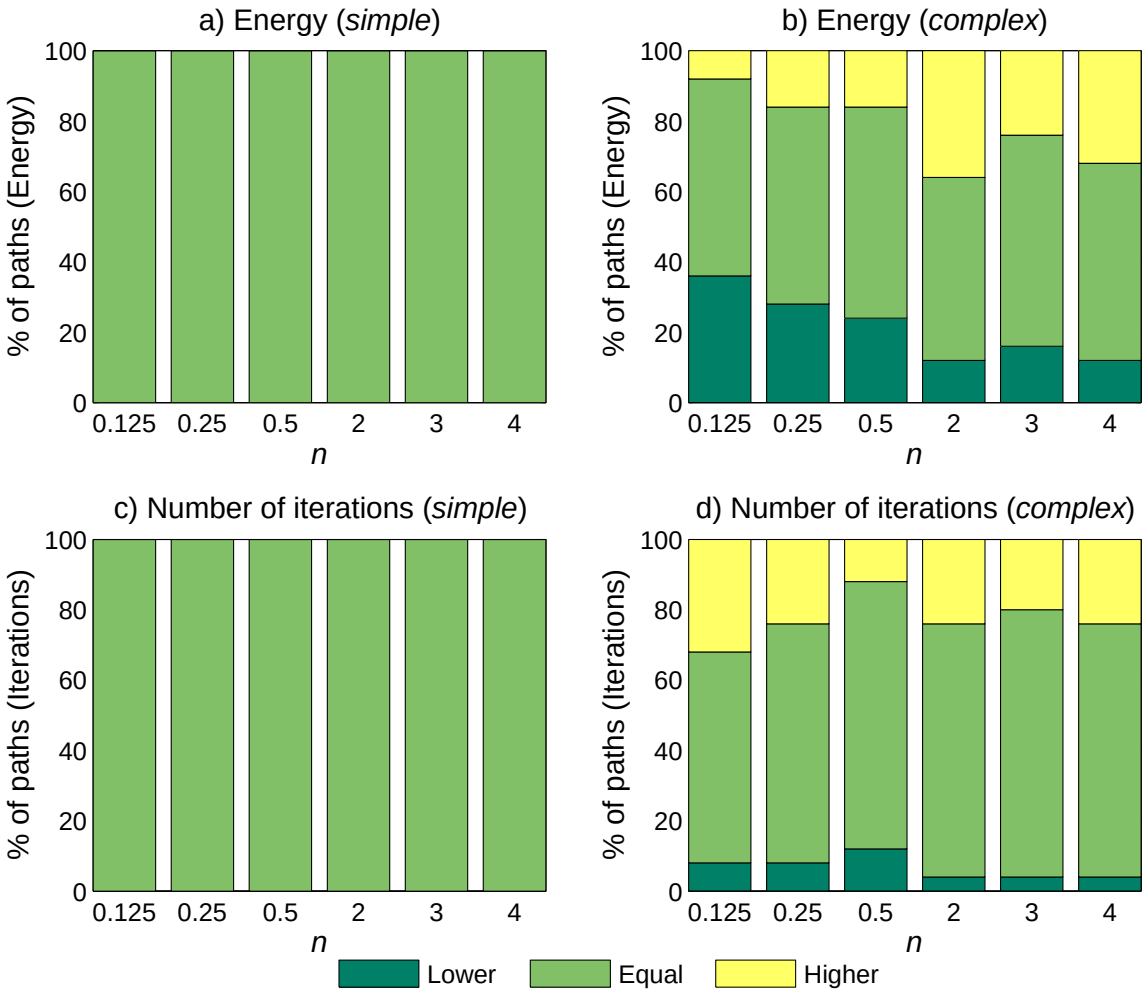

Figure 5.28: Newton-XEL (*Hessian only*): The percentage of paths that yields the lower, the equal, or the higher number of iterations in c) and d) or energy in a) and b) compared to $n = 1$. Shown in a) and c) are the percentage of paths on the simple energy landscape and in b) and d), on the complex energy landscape. Paths are generated by a bracketing and Brent line search.

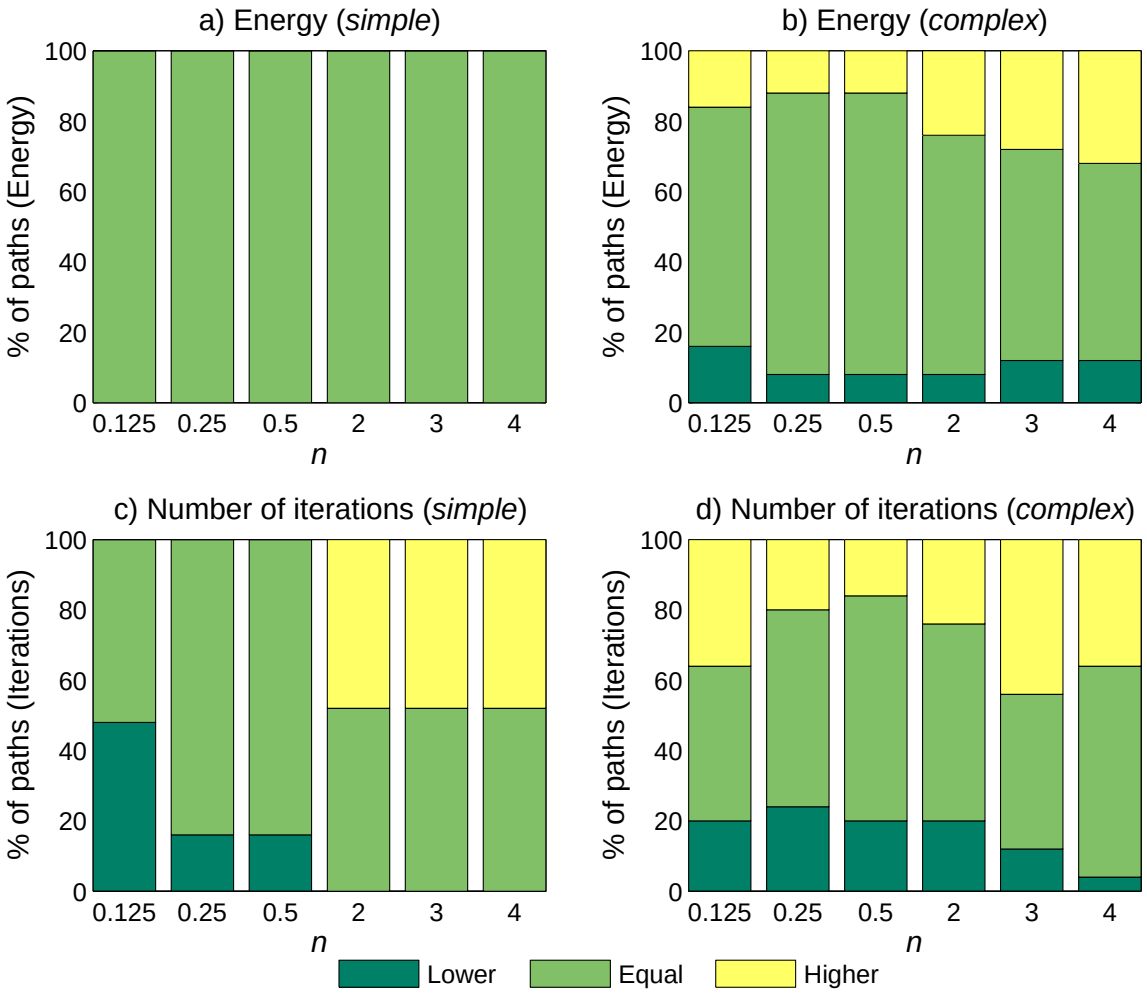

Figure 5.29: Newton-XEL: The percentage of paths that yields the lower, the equal, or the higher number of iterations in c) and d) or energy in a) and b) compared to $n = 1$. Shown in a) and c) are the percentage of paths on the simple energy landscape and in b) and d), on the complex energy landscape. Paths are generated by a heuristic exhaustive line search.

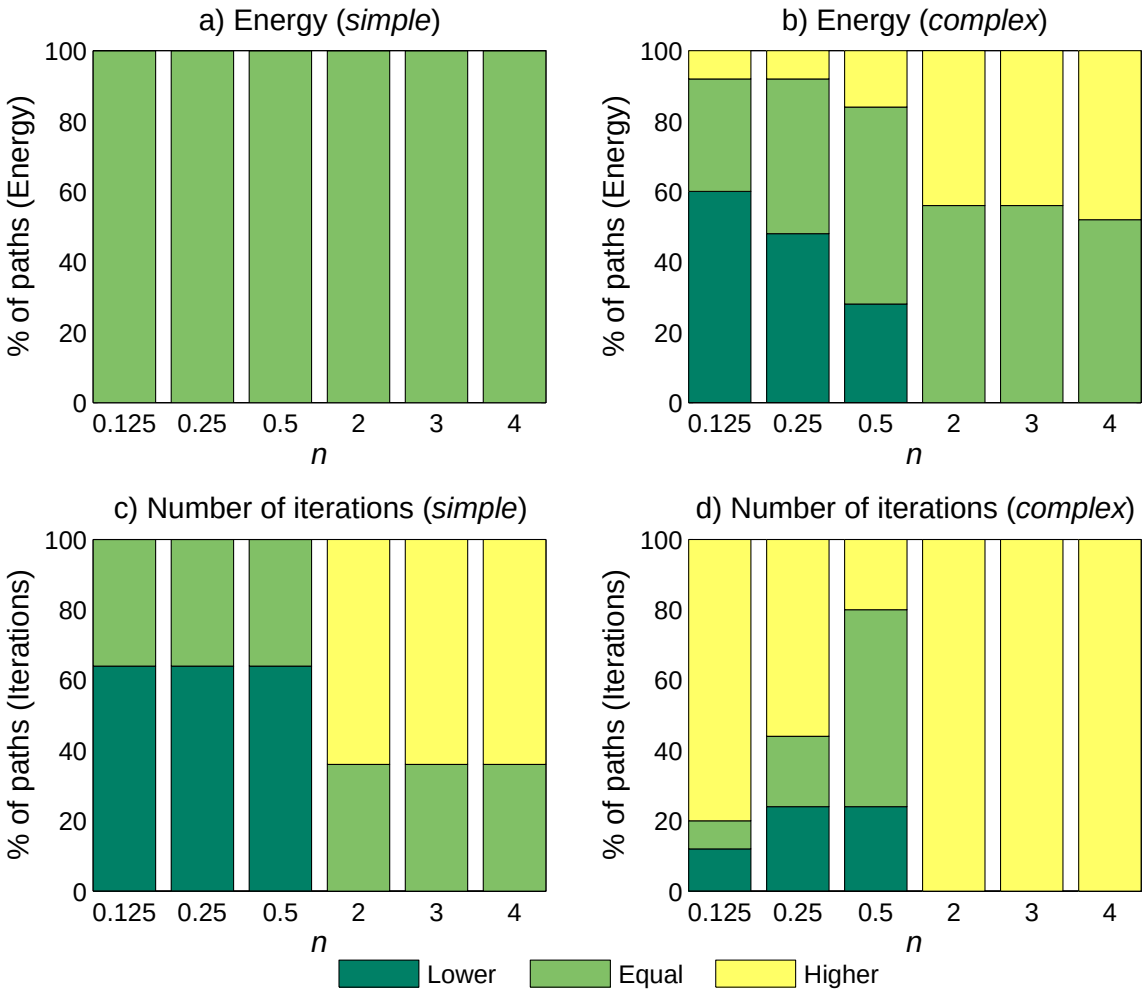

Figure 5.30: Newton-XEL (*Hessian only*): The percentage of paths that yields the lower, the equal, or the higher number of iterations in c) and d) or energy in a) and b) compared to $n = 1$. Shown in a) and c) are the percentage of paths on the simple energy landscape and in b) and d), on the complex energy landscape. Paths are generated by a heuristic exhaustive line search.

of the step of the Newton-XEL (*Hessian only*) than that of the Newton-XEL.

With a heuristic exhaustive line search, the Newton-XEL (Figure 5.29 a and c) on the simple energy landscape has the same quality for all n but higher speed for $n < 1$ and lower speed for $n > 1$. All paths have equal energy compared to $n = 1$ but 16% to 48% of $n < 1$ paths have a lower number of iterations while 48% of $n > 1$ paths have a higher number of iterations. The lower speed is due to the small step size of $n > 1$ cases. On the complex energy landscape (Figure 5.29 b and d) quality and speed become slightly worse. Although up to 16% of paths for all n cases from the Newton-XEL have lower energy than $n = 1$ and up to 24% have lower number of iterations, 12% to 32% have higher energy and 16% to 44% have higher number of iterations. The lower speed is again due to the small step size of $n > 1$ cases.

With a heuristic exhaustive line search, the Newton-XEL (*Hessian only*) (Figure 5.30 a and c) on the simple energy landscape have the same quality for all n but higher speed for $n < 1$ and lower speed for $n > 1$. All paths have equal energy compared to $n = 1$ but 64% of $n < 1$ paths have a lower number of iterations while 64% of $n > 1$ paths have a higher number of iterations. On the complex energy landscape (Figure 5.30 b and d) quality becomes better for $n < 1$ cases and worse for $n > 1$ and speed becomes significantly worse. As $n < 1$, 28% to 60% of paths have lower energy than $n = 1$ but as $n > 1$, 44% to 48% have higher energy. Their poorer performance is due to their smaller step size, which does not allow paths to travel beyond local minima. While some $n < 1$ paths have a lower number of iterations, 100% of the $n > 1$ paths have a higher number of iterations. While the low speed in the $n > 1$ paths is due to their small step size, the reduction in speed in $n < 1$ cases is due to their large step size and an increase in the complexity of the energy landscape. Since a larger step tends to overshoot a minimum, $n < 1$ paths take several iterations to reach a minimum as they keep passing it which is inefficient. This is less severe in the simple energy landscape because its global minimum is very flat. When a path

gets in the proximity of the global minimum, the convergence criterion is met and the path stops. Since the complex energy landscape is rougher than the simple energy landscape, the path needs to get much closer to the minumum before the convergence criterion is met.

The results show that Newton-XEL (*Hessian only*) are more dependent on n than those of Newton-XEL. Results also show that the bracketing and Brent line search may be a better choice of a line search because it can slightly improve quality without a substantial reduction of speed. While the heuristic exhaustive line search can achieve better quality and speed in some cases, it can also significantly worsen the quality and speed of the results.

5.1.2.2 QNA-XEL and QNA-XEL (Hessian only)

Figures 5.31 through 5.34 display the percentage of paths from QNA-XEL and QNA-XEL (*Hessian only*) with different n cases that has smaller, equal, or higher number of iterations or energy than those of $n = 1$ on the simple and the complex energy landscape. While paths resulting in Figures 5.31 and 5.32 are generated by a bracketing and Brent line search, paths resulting in Figures 5.33 and 5.34 are generated by a heuristic exhaustive line search. The higher percentage of paths with smaller energy or number of iterations implies better quality and speed.

With a bracketing and Brent line search, the QNA-XEL (Figure 5.31 a and c) on the simple energy landscape has slightly better quality for all n, lower speed for $n < 1$, and about the same speed for $n > 1$ compared to $n = 1$. As 8% to 16% of paths of most n cases have lower energy compared to $n = 1$, only a few n cases have 8% to 16% of paths with higher energy. In terms of speed 16% to 48% of paths for all n cases have higher number of iterations but up to 40% of paths with $n > 1$ have lower number of iterations. On the complex energy landscape (Figure 5.31 b and d) quality on average is about the same as $n = 1$ but speed becomes worse. On average

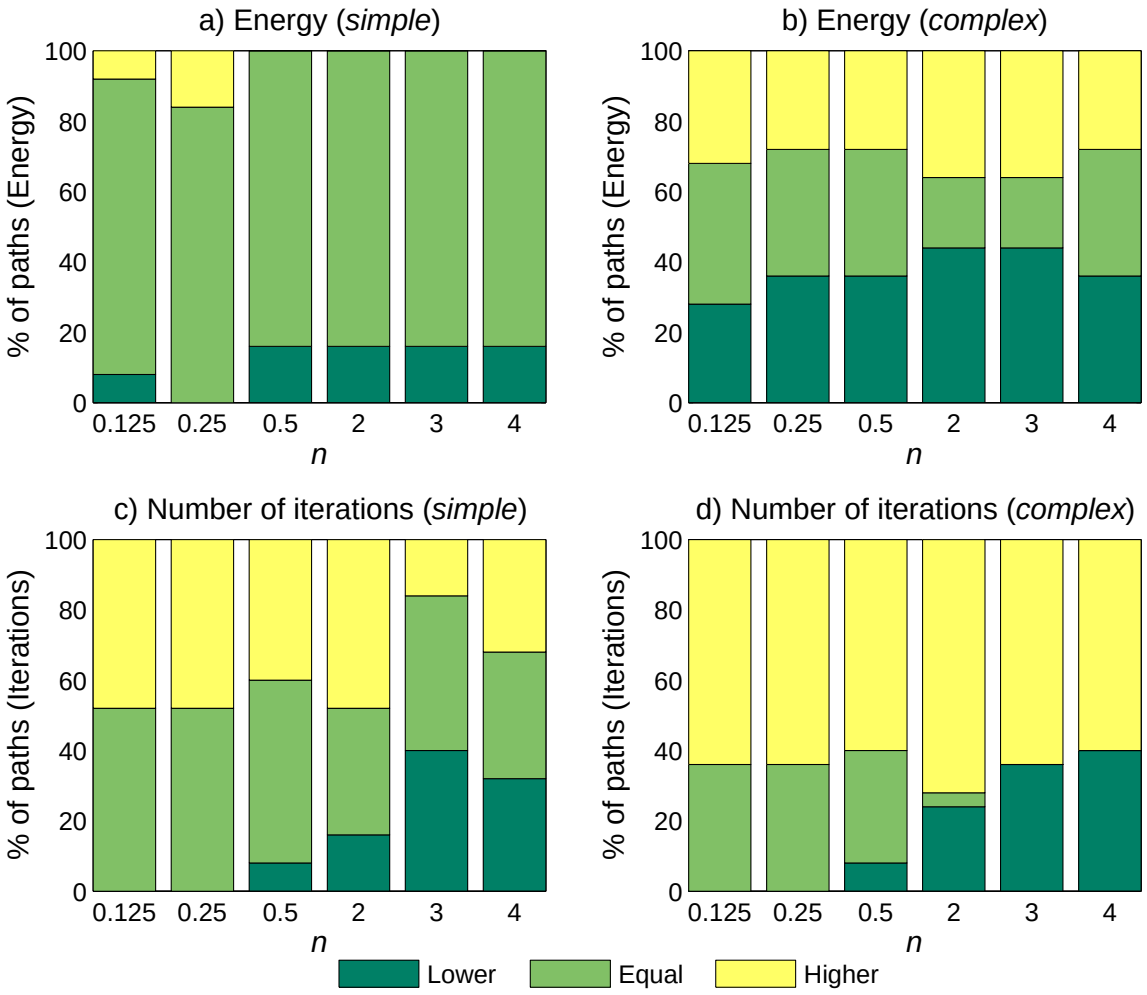

Figure 5.31: QNA-XEL: The percentage of paths that yields the lower, the equal, or the higher number of iterations in c) and d) or energy in a) and b) compared to $n = 1$. Shown in a) and c) are the percentage of paths on the simple energy landscape and in b) and d), on the complex energy landscape. Paths are generated by a bracketing and Brent line search.

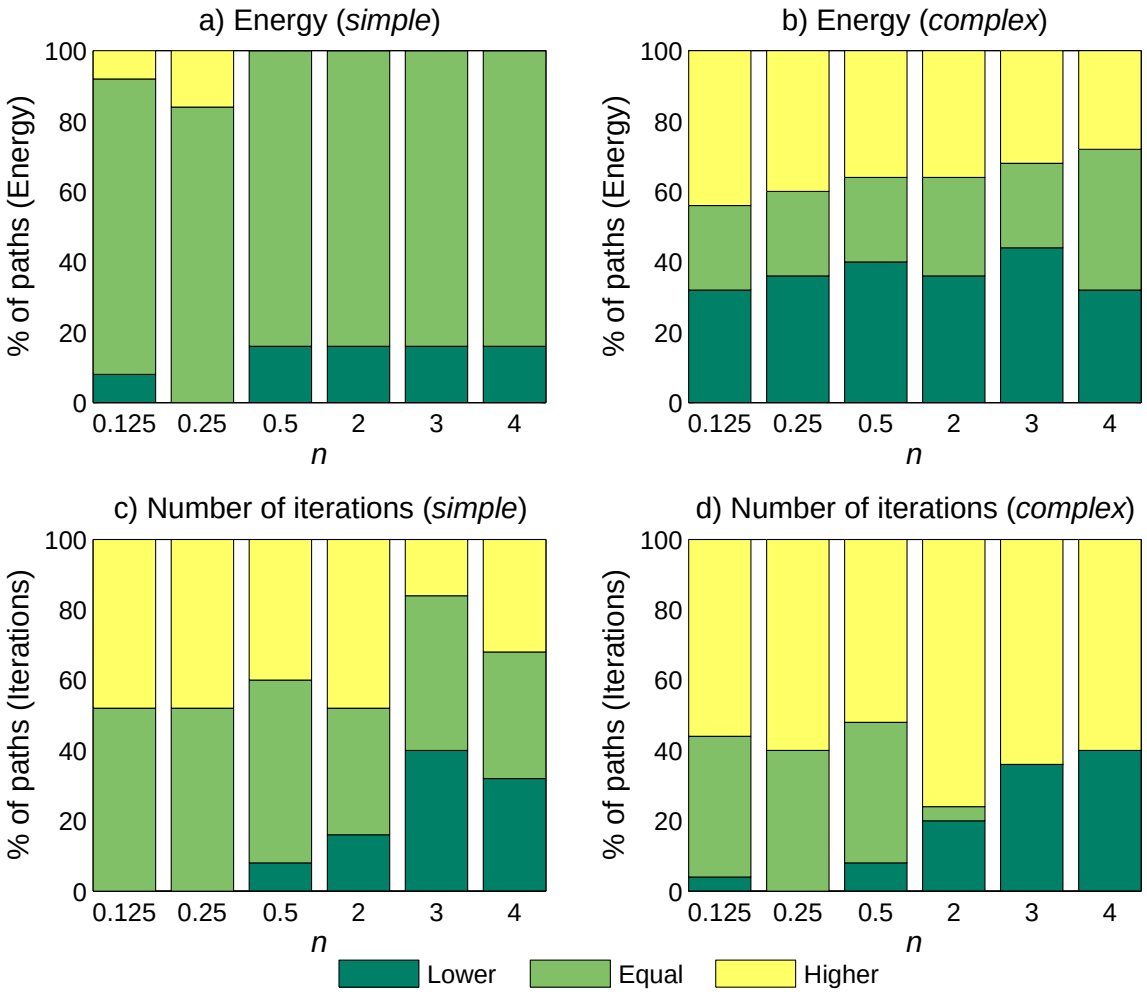

Figure 5.32: QNA-XEL (*Hessian only*): The percentage of paths that yields the lower, the equal, or the higher number of iterations in c) and d) or energy in a) and b) compared to $n = 1$. Shown in a) and c) are the percentage of paths on the simple energy landscape and in b) and d), on the complex energy landscape. Paths are generated by a bracketing and Brent line search.

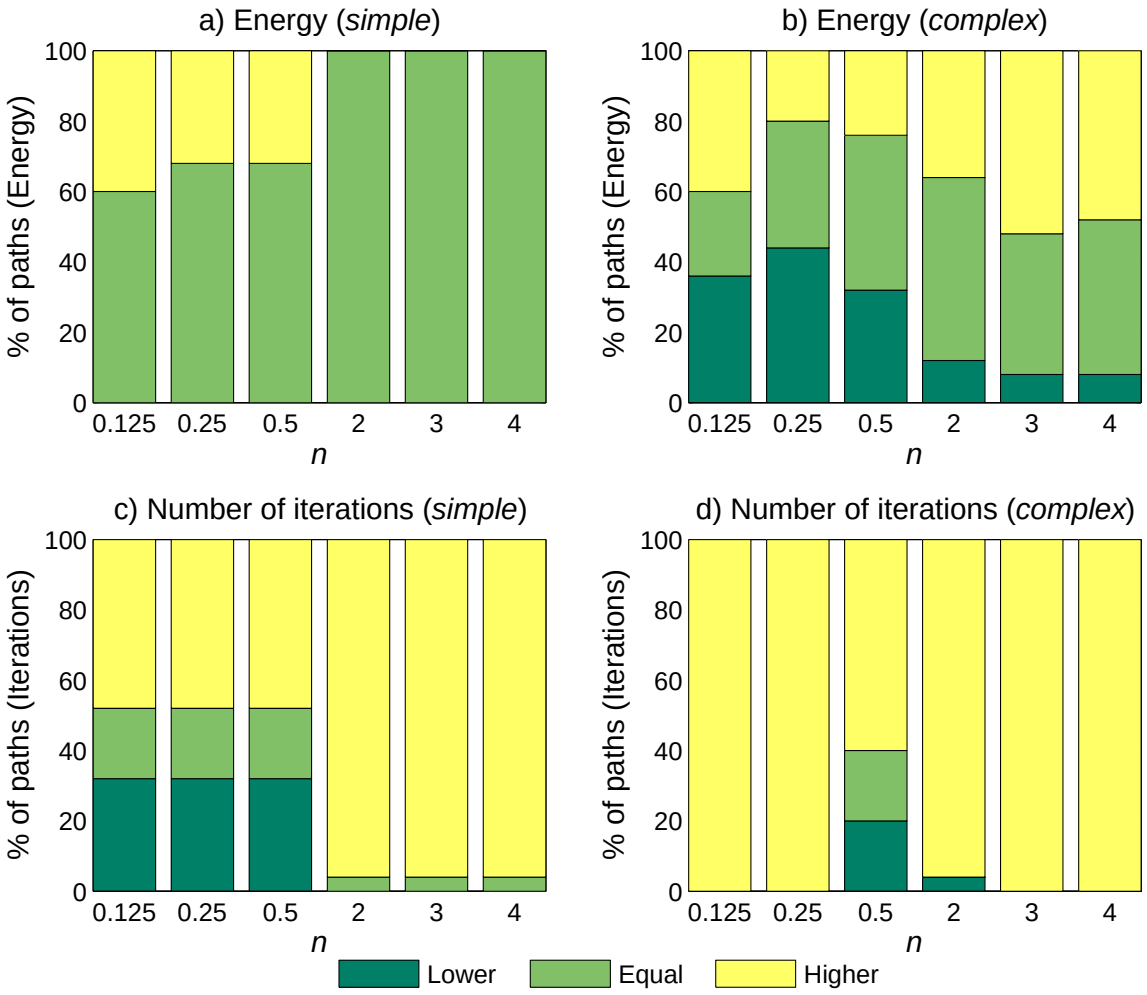

Figure 5.33: QNA-XEL: The percentage of paths that yields the lower, the equal, or the higher number of iterations in c) and d) or energy in a) and b) compared to $n = 1$. Shown in a) and c) are the percentage of paths on the simple energy landscape and in b) and d), on the complex energy landscape. Paths are generated by a heuristic exhaustive line search.

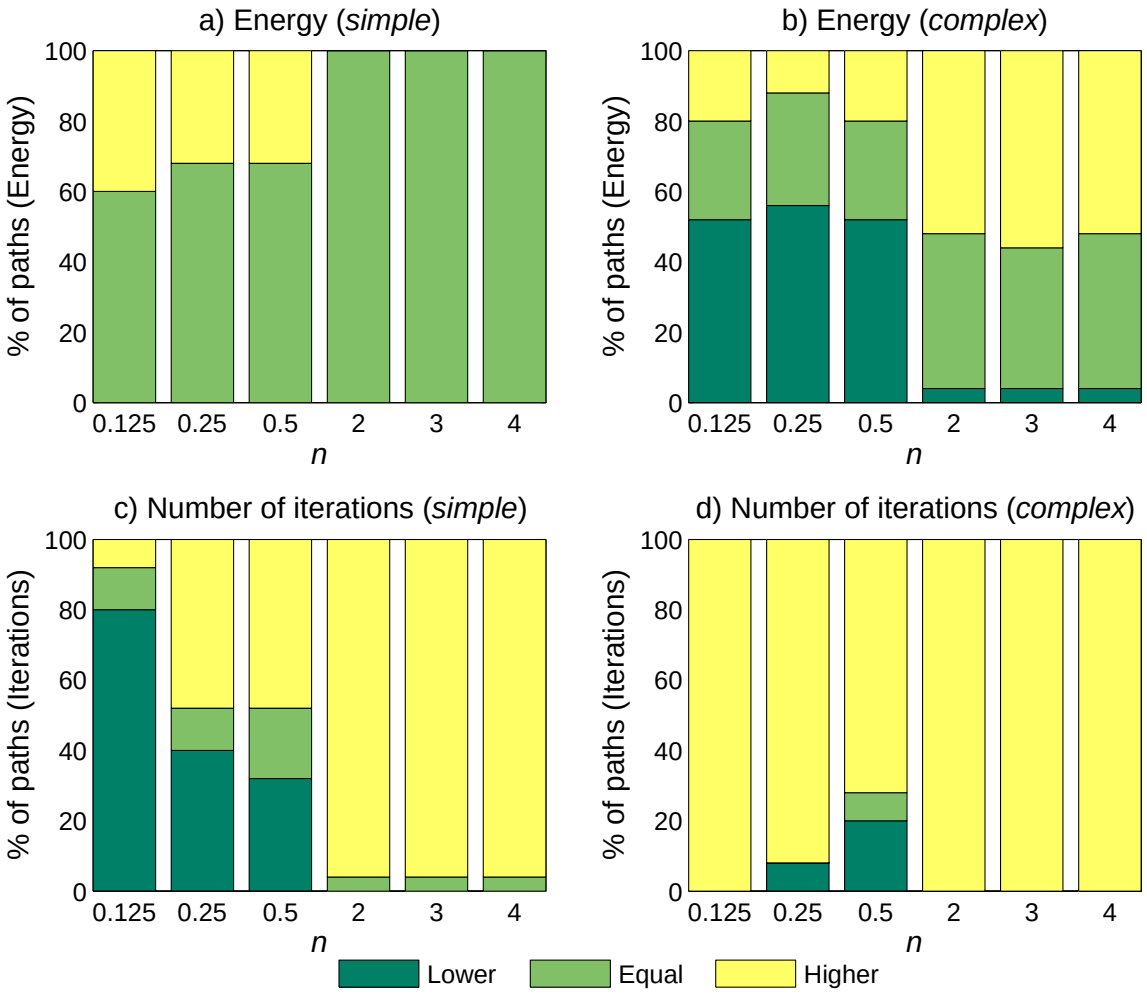

Figure 5.34: QNA-XEL (*Hessian only*): The percentage of paths that yields the lower, the equal, or the higher number of iterations in c) and d) or energy in a) and b) compared to $n = 1$. Shown in a) and c) are the percentage of paths on the simple energy landscape and in b) and d), on the complex energy landscape. Paths are generated by a heuristic exhaustive line search.

quality is about the same as $n = 1$ because up to 44% of paths yield lower energy but up to 36% of paths have higher energy. In terms of speed while up to 72% of all paths have a higher number of iterations, only 36% of $n > 1$ paths have a lower number of iterations. Results show that $n > 1$ yields slightly better quality and higher speed than $n < 1$ for QNA-XEL with a bracketing and Brent line search.

With a bracketing and Brent line search, results from the QNA-XEL (*Hessian only*) (Figure 5.32 a and c) are almost identical to those from the QNA-XEL which is consistent with observations made in the previous section. The QNA-XEL (*Hessian only*) on the simple energy landscape has slightly better quality for all n and lower speed for $n < 1$ and about the same speed for $n > 1$ compared to $n = 1$. On the complex energy landscape (Figure 5.32 b and d) quality on average is about the same as $n = 1$ but speed becomes worse. Results also show that $n > 1$ yields slightly better quality and higher speed than $n < 1$ for QNA-XEL (*Hessian only*) with a bracketing and Brent line search.

With a heuristic exhaustive line search, the QNA-XEL (Figure 5.33 a and c) on the simple energy landscape has the same quality for $n > 1$, slightly worse quality for $n < 1$, and lower speed for all n compared to $n = 1$. As 32% to 40% of $n < 1$ paths have higher energy than that of $n = 1$, all $n > 1$ paths have the same energy. This is because some $n < 1$ paths can not reach the global solution before the convergence criterion is met. Due to an inaccurate estimated modified Hessian matrix $\hat{K}^*$ and a very small step size, $\hat{K}^*$ does not get improved so the paths are almost perpendicular to the gradient and advance with very small steps. It is possible that if the convergence criterion becomes stricter, these paths may eventually reach the global minimum. In terms of speed, while only 32% of $n < 1$ paths have a lower number of iterations, 48% of $n < 1$ paths and 96% of $n > 1$ paths have a higher number of iterations.

With a heuristic exhaustive line search, the QNA-XEL on the complex energy landscape (Figure 5.33 b and d) has the same quality for $n < 1$ but worse quality for

$n > 1$ and lower speed. While up to 40% of $n < 1$ paths have higher energy, up to 44% have lower energy so on average the quality is about the same as $n = 1$. For $n > 1$ the quality is worse as up to 52% of the paths have higher energy than $n = 1$. Speed is significantly worse than $n = 1$ as 60% to 100% of all paths have a higher number of iterations. The lower speed is due to inaccuracy of $\hat{K}^*$ in $n < 1$ cases and too small step size in $n > 1$ cases.

With a heuristic exhaustive line search, results from QNA-XEL (*Hessian only*) (Figure 5.34 a and c) are considerably similar to those from QNA-XEL. On the simple energy landscape QNA-XEL (*Hessian only*) has the same quality compared to $n = 1$ for $n > 1$ but slightly worse quality for $n < 1$ and lower speed for all n except for $n = 0.125$. On the complex energy landscape (Figure 5.34 b and d) it has slightly better quality for $n < 1$ but worse quality for $n > 1$ and lower speed.

Results show that the heuristic exhaustive line search can significantly reduce the quality and the speed of QNA-XEL and QNA-XEL (*Hessian only*).

5.1.2.3 BFGS-XEL and BFGS-XEL (Hessian only)

Figures 5.35 through 5.38 display the percentage of paths from BFGS-XEL and BFGS-XEL (*Hessian only*) with different n cases that have smaller, equal, or higher number of iterations or energy than those of $n = 1$ on the simple and the complex energy landscape. While paths resulting in Figures 5.35 and 5.36 are generated by a bracketing and Brent line search, paths resulting in Figures 5.37 and 5.38 are generated by a heuristic exhaustive line search. The higher percentage of paths with a lower energy or a smaller number of iterations implies better quality or speed.

With a bracketing and Brent line search, the BFGS-XEL (Figure 5.35 a and c) on the simple energy landscape has the same quality and speed for all n compared to $n = 1$. In terms of quality all paths have equal energy compared to $n = 1$. In terms of speed 8% to 24% of most paths have a higher number of iterations but the

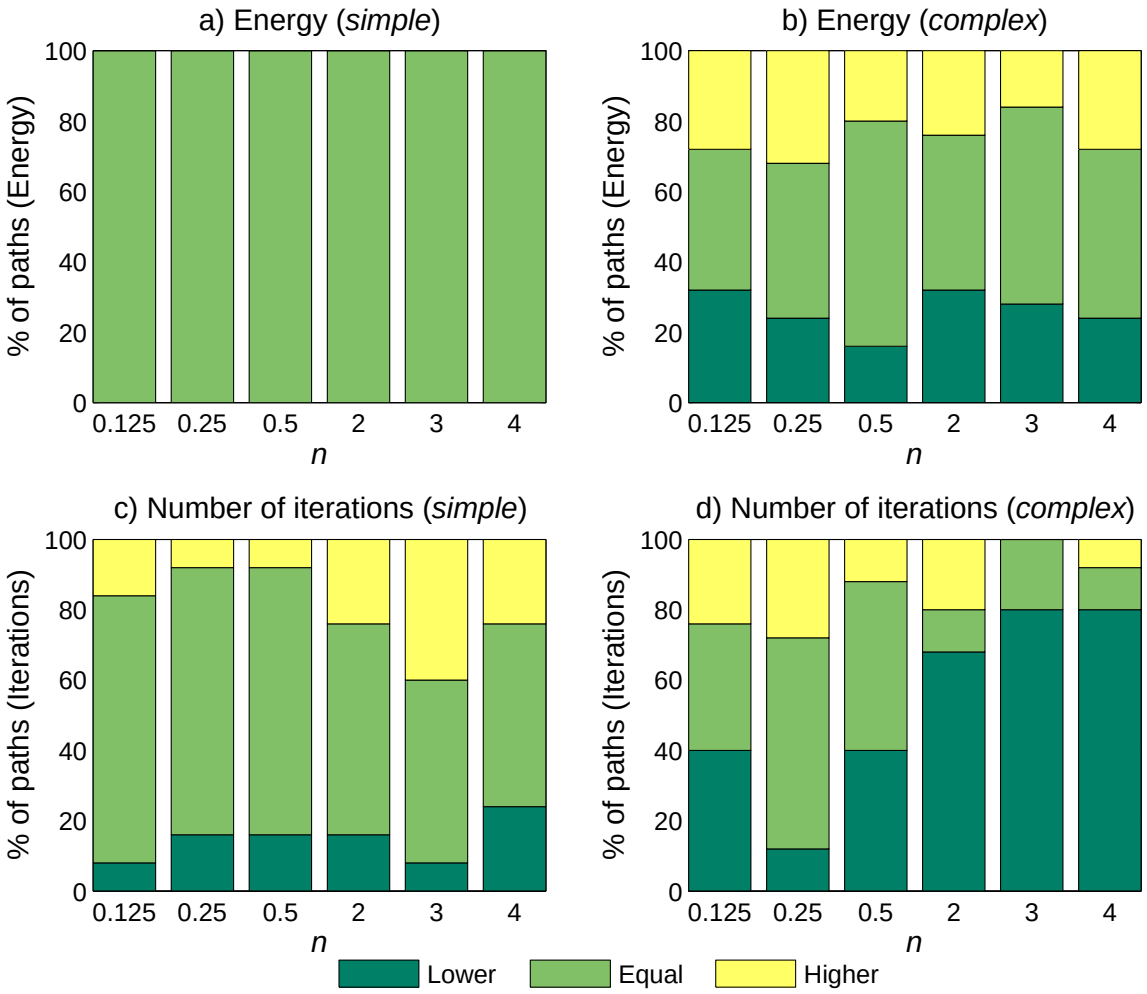

Figure 5.35: BFGS-XEL: The percentage of paths that yields the lower, the equal, or the higher number of iterations in c) and d) or energy in a) and b) compared to $n = 1$. Shown in a) and c) are the percentage of paths on the simple energy landscape and in b) and d), on the complex energy landscape. Paths are generated by a bracketing and Brent line search.

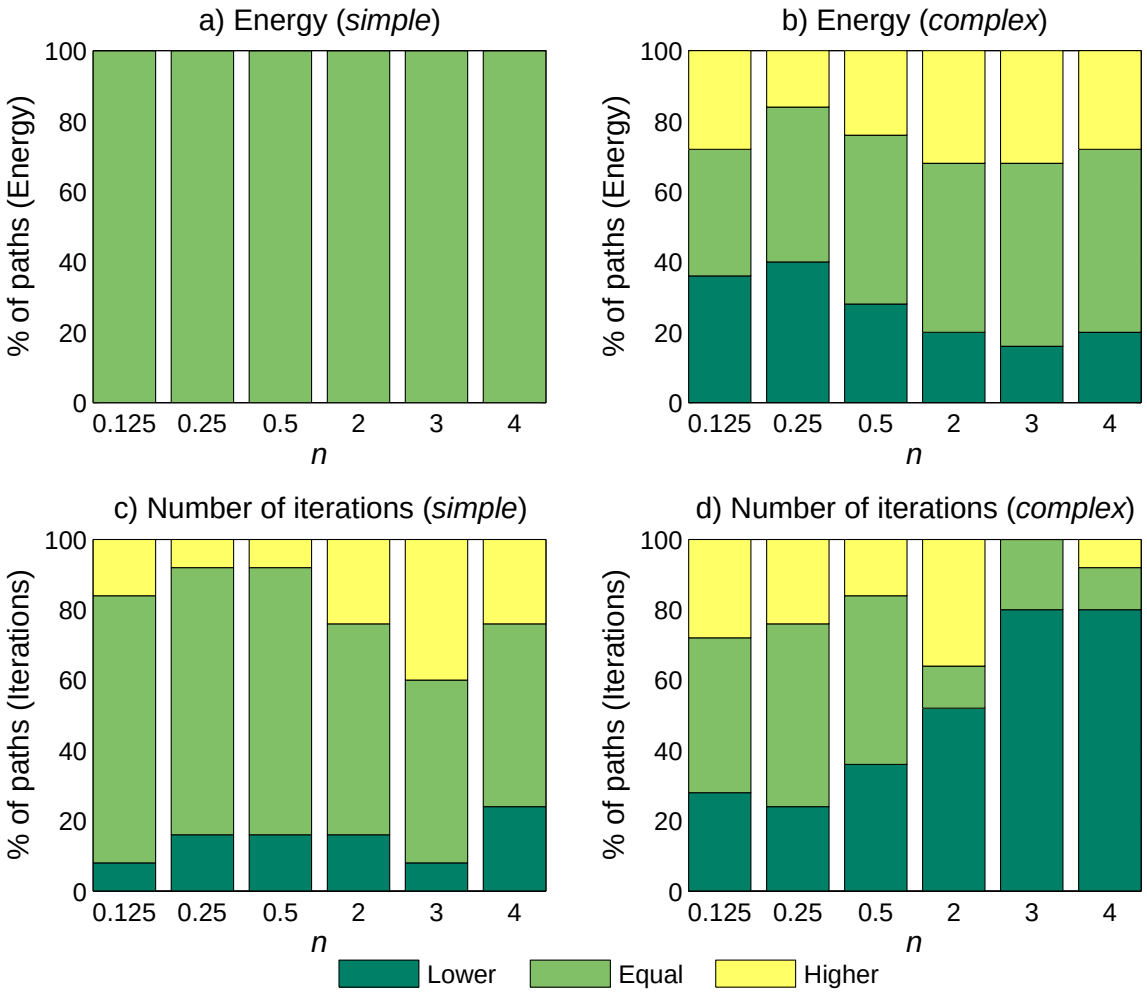

Figure 5.36: BFGS-XEL (*Hessian only*): The percentage of paths that yields the lower, the equal, or the higher number of iterations in c) and d) or energy in a) and b) compared to $n = 1$. Shown in a) and c) are the percentage of paths on the simple energy landscape and in b) and d), on the complex energy landscape. Paths are generated by a bracketing and Brent line search.

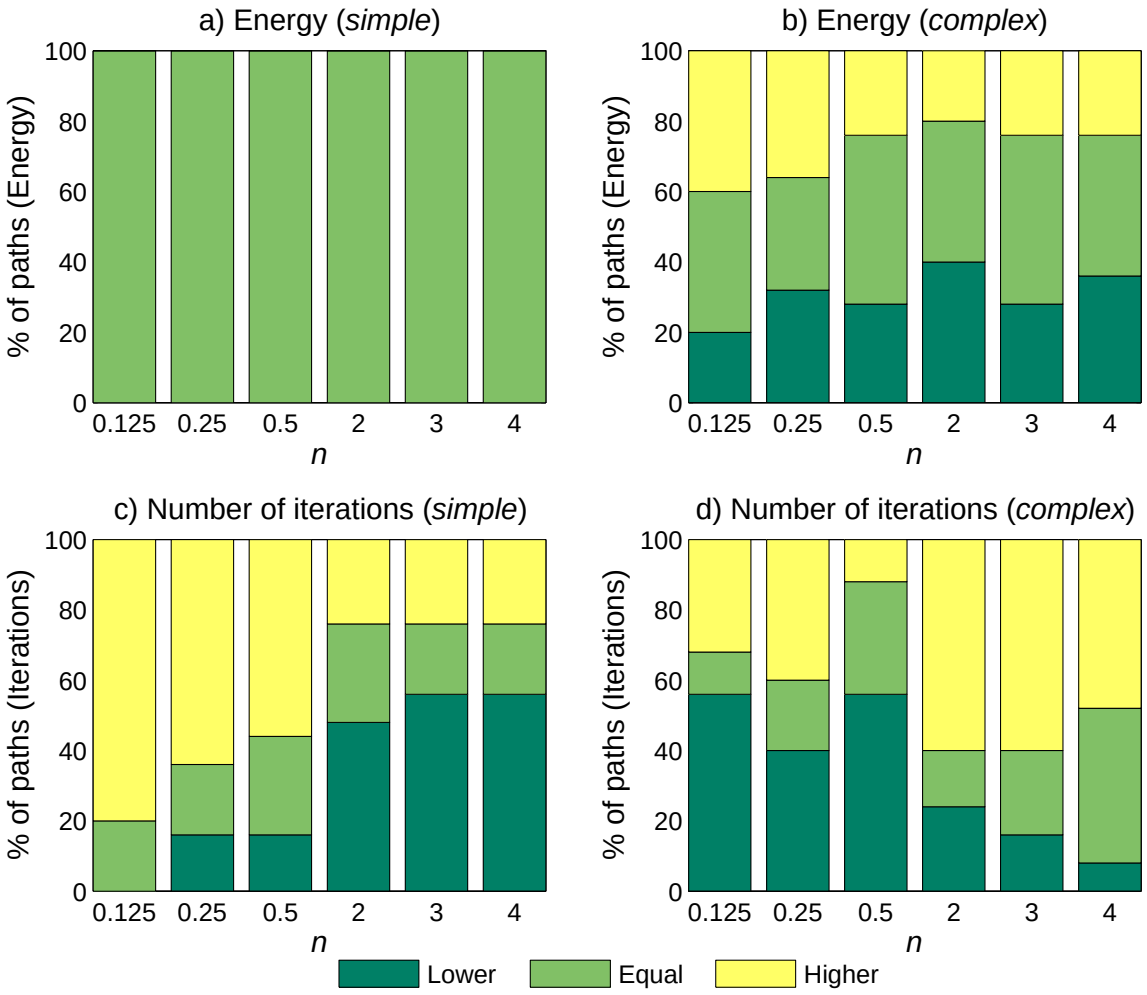

Figure 5.37: BFGS-XEL: The percentage of paths that yields the lower, the equal, or the higher number of iterations in c) and d) or energy in a) and b) compared to $n = 1$. Shown in a) and c) are the percentage of paths on the simple energy landscape and in b) and d), on the complex energy landscape. Paths are generated by a heuristic exhaustive line search.

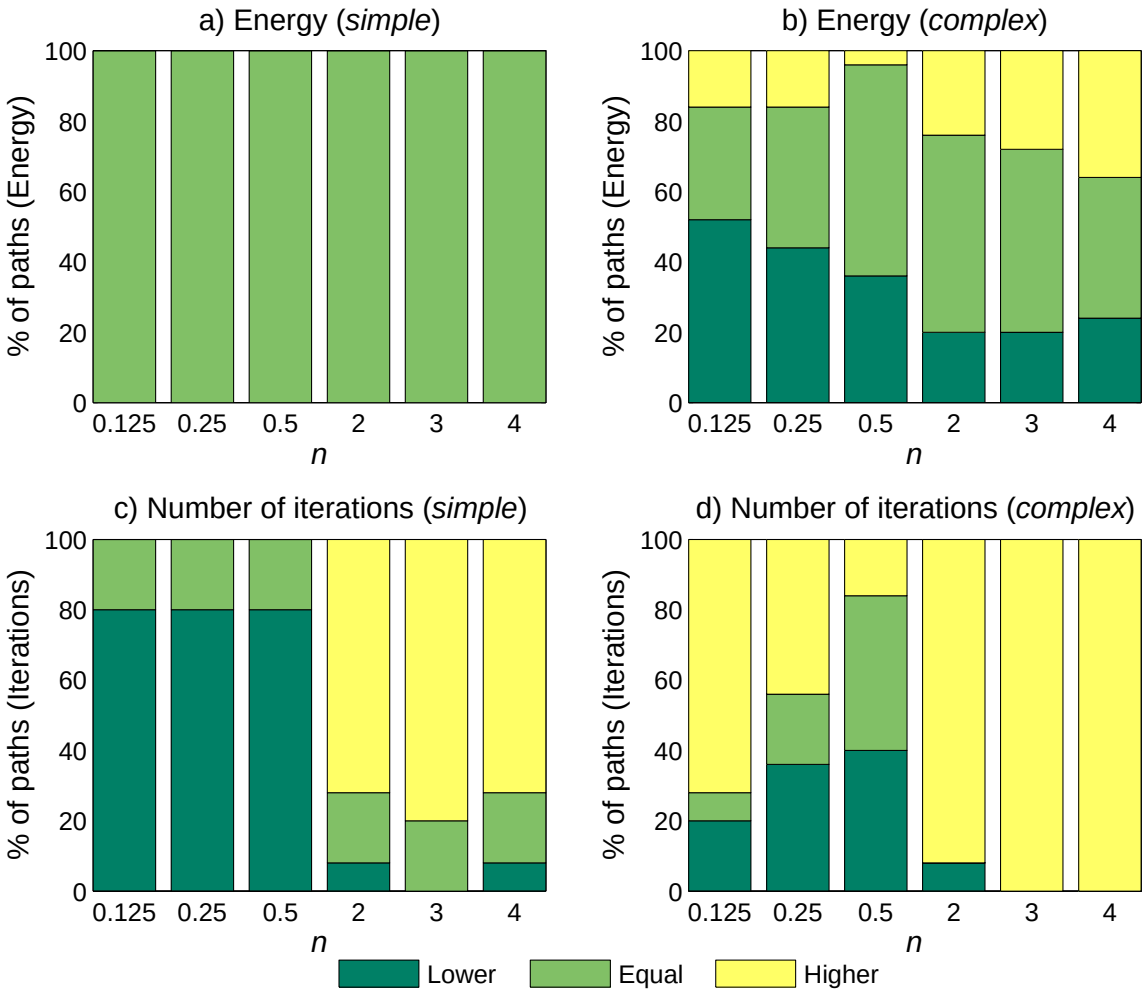

Figure 5.38: BFGS-XEL (*Hessian only*): The percentage of paths that yields the lower, the equal, or the higher number of iterations in c) and d) or energy in a) and b) compared to $n = 1$. Shown in a) and c) are the percentage of paths on the simple energy landscape and in b) and d), on the complex energy landscape. Paths are generated by a heuristic exhaustive line search.

same paercentage have lower number of iterations. On the complex energy landscape (Figure 5.35 b and d) quality on average is about the same as $n = 1$ but speed is better. In terms of quality up to 32% of all paths have lower energy but the same approximate percentage of paths have higher energy. In terms of speed while up to only 28% of all paths have a higher number of iterations, up to 40% of $n < 1$ paths and up to 80% of $n > 1$ paths have a lower number of iterations. Results show that $n > 1$ can yield significantly higher speed than $n < 1$ for BFGS-XEL with a bracketing and Brent line search.

With a bracketing and Brent line search, results from the BFGS-XEL (*Hessian only*) (Figure 5.36 a and c) are substantially similar to those from the BFGS-XEL which is consistent with observations made in the previous section. The BFGS-XEL (*Hessian only*) on the simple energy landscape has the same quality and speed for all n compared to $n = 1$. On the complex energy landscape (Figure 5.36 b and d) on average quality is about the same as $n = 1$ but speed is better. Results show that $n < 1$ yields slightly better quality than $n > 1$ while $n > 1$ yields higher speed than $n < 1$ for BFGS-XEL (*Hessian only*) with a bracketing and Brent line search.

With a heuristic exhaustive line search, the BFGS-XEL (Figure 5.37 a and c) on the simple energy landscape has the same quality for all n compared to $n = 1$, lower speed for $n < 1$, and higher speed for $n > 1$. In terms of quality all paths have equal energy compared to $n = 1$. In terms of speed up to 56% of $n > 1$ paths have a lower number of iterations but only 24% have higher number of iterations. For $n < 1$ up to 80% have higher number of iterations. The low speed in $n < 1$ is caused by the inaccuracy of the estimated modified Hessian inverse $\hat{H}^*$ so $n < 1$ paths take longer to converge. On the complex energy landscape (Figure 5.37 b and d) the BFGS-XEL have about the same quality on average but lower speed for $n > 1$ and higher speed for $n < 1$. While up to 40% of all paths have either higher or lower energy so on average the quality is about the same as $n = 1$. In terms of speed up to 60% of the

paths have higher energy than $n = 1$ but up to 56% of $n < 1$ paths have a lower number of iterations. The effect of n on speed on the complex energy landscape is opposite to that on the simple energy landscape.

With a heuristic exhaustive line search, the BFGS-XEL (*Hessian only*) (Figure 5.38 a and c) on the simple energy landscape has the same quality for all n compared to $n = 1$, lower speed for $n > 1$, and higher speed for $n < 1$. In terms of quality all paths have equal energy compared to $n = 1$. In terms of speed up to 80% of $n > 1$ paths have a higher number of iterations but only 8% have higher number of iterations. For $n < 1$ up to 80% have lower number of iterations. The low speed in $n > 1$ is possibly caused by the step size being too small. On the complex energy landscape (Figure 5.38 b and d) the BFGS-XEL (*Hessian only*) have better quality for $n < 1$ but slightly worse quality for $n > 1$ and lower speed. In terms of quality up to 52% of $n < 1$ paths have lower energy but up to only 16% have higher energy and up to 36% of $n > 1$ paths have higher energy but up to only 24% have lower energy. In terms of speed $n = 0.5$ yields higher speed but up to 100% of paths in most n cases have a higher number of iterations. The results show that $n < 1$ yields better results than $n > 1$ for the BFGS-XEL (*Hessian only*) with a heuristic exhaustive line search.

The bracketing and Brent line search yields more consistent results for different values of n than the heuristic exhaustive line search and it yields better quality and speed on average.

5.1.2.4 Conclusion of iterations and energy comparison to the unmodified case

The energy and the number of iterations for each path generated by Newton-XEL, Newton-XEL (*Hessian only*), QNA-XEL, QNA-XEL (*Hessian only*), BFGS-XEL, and BFGS-XEL (*Hessian only*) with different values of n are compared to those of $n = 1$. The results are the percentage of paths with smaller, equal, or higher energy and number of iterations are presented from which the quality and the speed of each algorithm are evaluated. Tables 5.2 and 5.3 summarize the quality and the speed of XEL and XEL (*Hessian only*) algorithms with $n < 1$ and $n > 1$. Quality or speed is "Significantly better" when the averaged percentage of paths with lower energy or number of iterations is $\geq 50\%$ higher than the averaged percentage of paths with higher energy or number of iterations, "Better" when it is $\geq 30\%$ higher, and "Slightly better" when it is $\geq 10\%$ higher. Quality or speed is "Significantly worse" when the averaged percentage of paths with lower energy or number of iterations is $\geq 50\%$ lower than the averaged percentage of paths with higher energy or number of iterations, "Worse" when it is $\geq 30\%$ lower, "Slightly worse" when it is $\geq 10\%$ lower. Otherwise quality or speed is "Same."

Results in Tables 5.2 and 5.3 show that the XEL and the XEL (*Hessian only*) algorithms yield the same quality on the simple energy landscape which demonstrates that the difference in the magnitudes of the step of these algorithms does not effect the quality on a smooth landscape.

The results are inconclusive for which value of n yields better performance for all algorithms. While $n > 1$ yields better quality on the simple energy landscape, $n < 1$ yields better quality on the complex energy landscape. Although in each algorithm different values of n yield higher speed, $n > 1$ yields higher speed when used with the bracketing and Brent line search and $n < 1$ yields higher speed when used with the heuristic exhaustive line search. However, on average the bracketing and Brent line search performs better than the heuristic exhaustive line search.

On the complex energy landscape the BFGS-XEL and the BFGS-XEL (*Hessian only*) with the bracketing and Brent line search yield the best overall quality and speed without sacrificing one or another. Thus, they would be the methods of choice at least for these results. The fact that they perform well on the complex energy landscape can imply that they are more robust than the other algorithms and therefore they are most likely to perform better in the real energy landscape.

Table 5.2: A summary of the quality and the speed results: XEL algorithms. (1 of 2)

	Characteristic		Newton-XEL		QNA-XEL		BFGS-XEL	
			Brent	**Exhaustive**	**Brent**	**Exhaustive**	**Brent**	**Exhaustive**
Simple Energy Landscape	Quality	$n < 1$	Same	Same	Same	Worse	Same	Same
		$n > 1$	Same	Same	Slightly better	Same	Same	Same
	Speed	$n < 1$	Same	Slightly better	Worse	Slightly worse	Same	Significantly worse
		$n > 1$	Same	Worse	Same	Significantly worse	Slightly worse	Slightly better
Complex Energy Landscape	Quality	$n < 1$	Same	Same	Same	Same	Same	Same
		$n > 1$	Slightly worse	Slightly worse	Same	Worse	Same	Slightly better
	Speed	$n < 1$	Same	Same	Significantly worse	Significantly worse	Same	Slightly better
		$n > 1$	Slightly worse	Slightly worse	Worse	Significantly worse	Significantly better	Worse

Table 5.3: A summary of the quality and the speed results: XEL (*Hessian only*) algorithms. (2 of 2)

	Characteristic		Newton-XEL (*Hessian only*)		QNA-XEL (*Hessian only*)		BFGS-XEL (*Hessian only*)	
			Brent	**Exhaustive**	**Brent**	**Exhaustive**	**Brent**	**Exhaustive**
Simple Energy Landscape	Quality	$n < 1$	Same	Same	Same	Worse	Same	Same
		$n > 1$	Same	Same	Slightly better	Same	Same	Same
	Speed	$n < 1$	Same	Significantly better	Worse	Slightly better	Same	Significantly better
		$n > 1$	Same	Significantly worse	Same	Significantly worse	Slightly worse	Significantly worse
Complex Energy Landscape	Quality	$n < 1$	Slightly better	Better	Same	Better	Slightly better	Better
		$n > 1$	Slightly worse	Worse	Same	Worse	Slightly worse	Same
	Speed	$n < 1$	Slightly worse	Worse	Significantly worse	Significantly worse	Same	Slightly worse
		$n > 1$	Slightly worse	Significantly worse	Worse	Significantly worse	Significantly better	Significantly worse

5.1.3 Global Search Performance

Since Newton's method, the QNA, and the BFGS algorithm are local optimization algorithms, XEL algorithms are not expected to solve the global minimization. However, the XEL may improve the performance in a global search by transforming the energy landscape such that it assists the optimization. Here the global search performance is defined as an ability to locate the global solution. This ability contributes to the quality because a more successful global search yields a lower energy which implies higher quality. To compare the global search performance the number of paths that are able to locate the global minimum from Newton-XEL, Newton-XEL (*Hessian only*), QNA-XEL, QNA-XEL (*Hessian only*), BFGS-XEL, and BFGS-XEL (*Hessian only*) are compared. Figures 5.39 and 5.41 present results from XEL algorithms and Figures 5.40 and 5.42 present results from XEL (*Hessian only*) algorithms. Results in Figures 5.39 and 5.40 are generated by a bracketing and Brent line search, results in Figures 5.41 and 5.42 are generated by a heuristic exhaustive line search. Higher number of paths may indicate higher ability to locate the global minimum which is desired.

With a bracketing and Brent line search, every path generated by the Newton-XEL and the BFGS-XEL on the simple energy landscape (Figure 5.39 a) successfully locates the global minimum, but some n cases of QNA-XEL do not. Note that Newton-XEL has only 21 paths because some starting points have zero Hessian matrix and yield no path. High number of paths can locate the global minimum because of the simplicity of the energy landscape. On the complex energy landscape (Figure 5.39 b) far fewer paths from all algorithms can locate the global minimum. All algorithms where $n \neq 1$ have fewer successful paths than $n = 1$ and in QNA-XEL and BFGS-XEL $n > 1$ cases have more successful paths than $n < 1$.

With a bracketing and Brent line search, the results of the Newton-XEL (*Hessian only*), the QNA-XEL (*Hessian only*), and the BFGS-XEL (*Hessian only*) on the simple

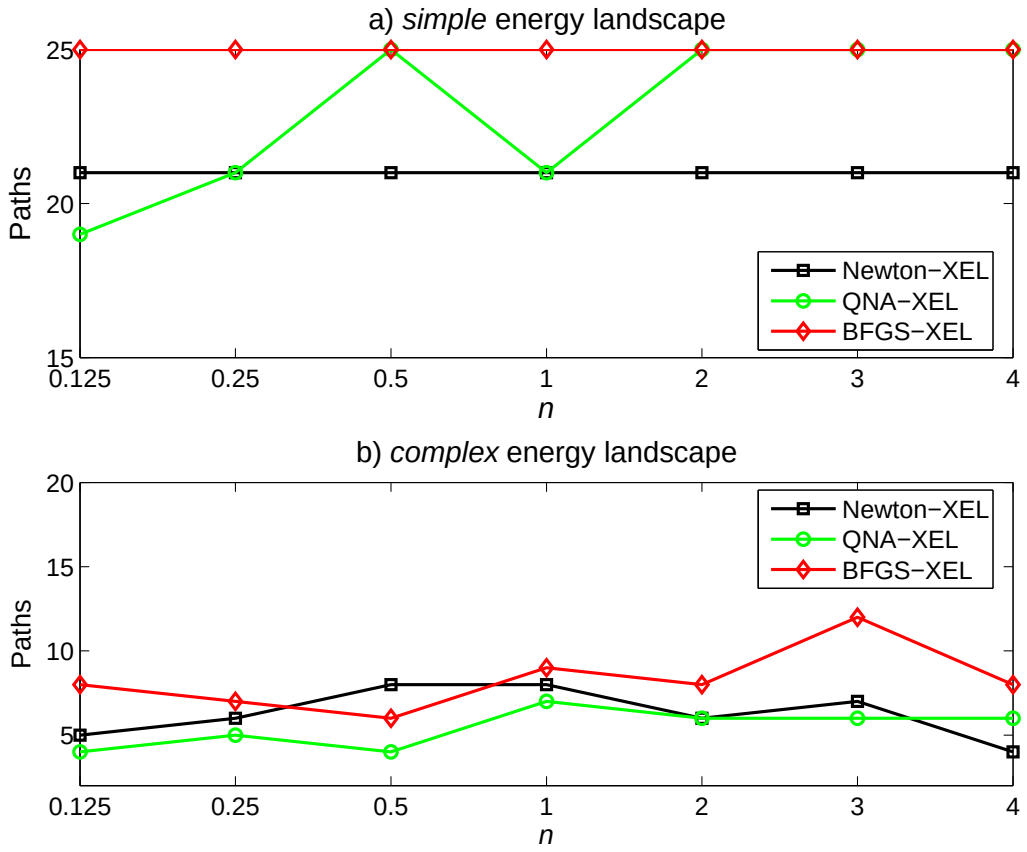

Figure 5.39: The number of paths on a simple or a complex energy landscape generated by Newton-XEL, QNA-XEL, or BFGS-XEL that locate the global minimum. Paths are generated by a bracketing and Brent line search.

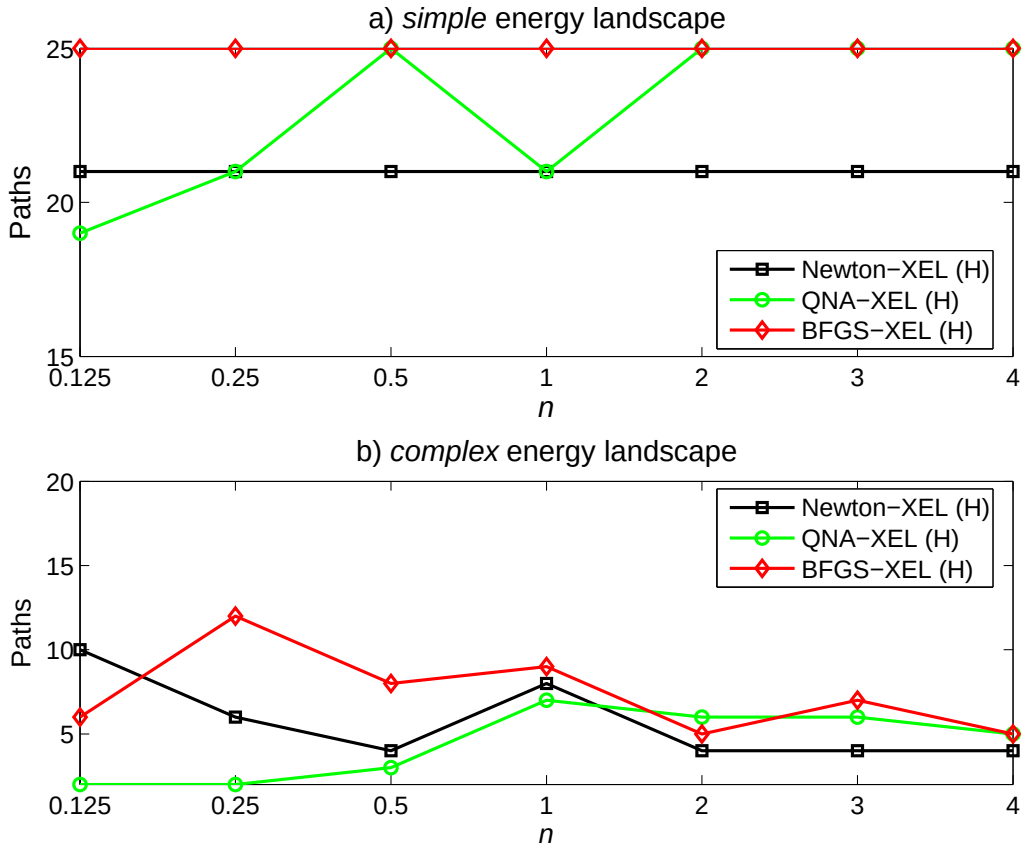

Figure 5.40: The number of paths on a simple or a complex energy landscape generated by Newton-XEL (*Hessian only*), QNA-XEL (*Hessian only*), or BFGS-XEL (*Hessian only*) that locate the global minimum. Paths are generated by a bracketing and Brent line search.

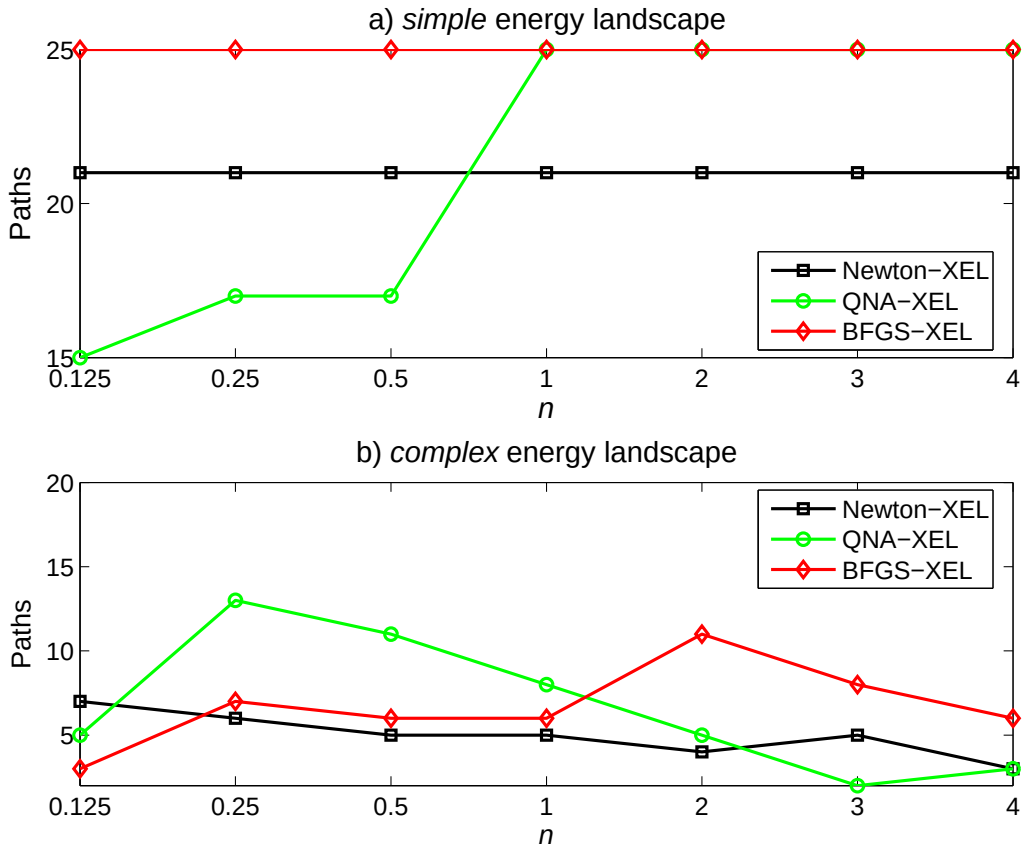

Figure 5.41: The number of paths on a simple or a complex energy landscape generated by Newton-XEL, QNA-XEL, or BFGS-XEL that locate the global minimum. Paths are generated by a heuristic exhaustive line search.

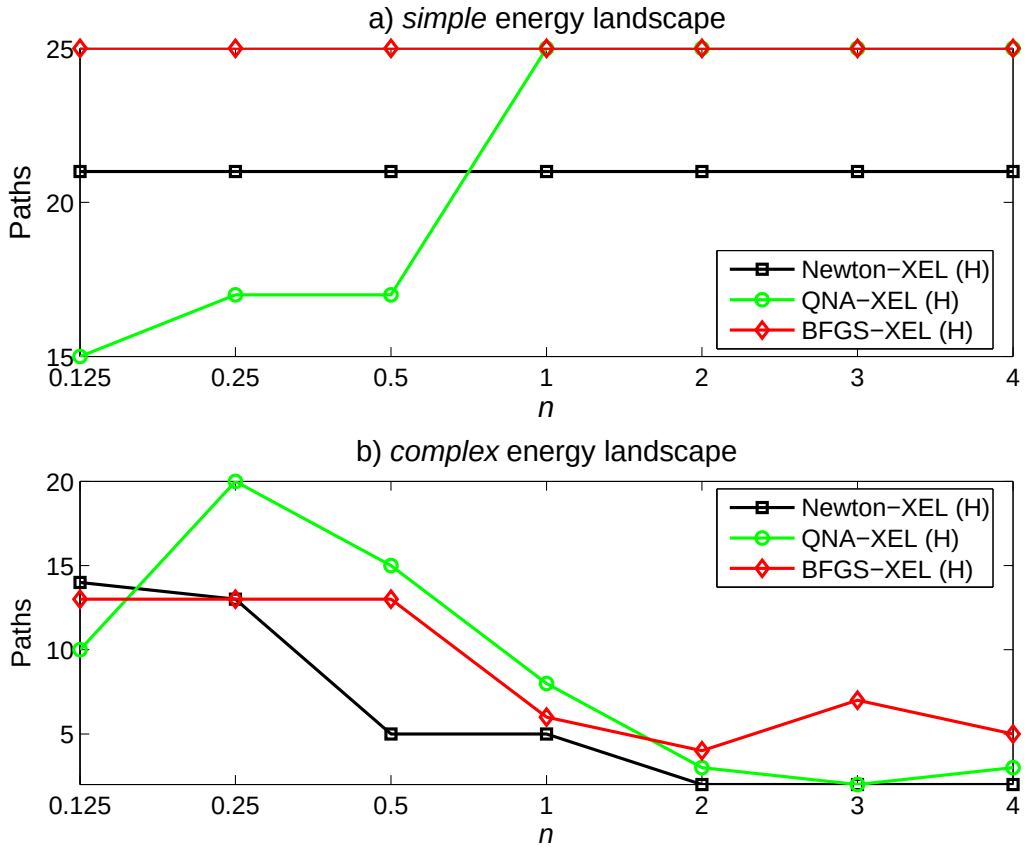

Figure 5.42: The number of paths on a simple or a complex energy landscape generated by Newton-XEL (*Hessian only*), QNA-XEL (*Hessian only*), or BFGS-XEL (*Hessian only*) that locate the global minimum. Paths are generated by a heuristic exhaustive line search.

energy landscape (Figure 5.40 a) are the same as those of the Newton-XEL, the QNA-XEL, and the BFGS-XEL. On the complex energy landscape (Figure 5.40 b) far fewer paths from all algorithms can locate the global minimum. All algorithms where $n \neq 1$ have fewer successful paths than $n = 1$ and in Newton-XEL (*Hessian only*) and BFGS-XEL (*Hessian only*) $n < 1$ cases have more successful paths than $n > 1$ but in QNA-XEL (*Hessian only*) $n > 1$ cases have more successful paths than $n < 1$.

With a heuristic exhaustive line search, every path generated by the Newton-XEL and the BFGS-XEL on the simple energy landscape (Figure 5.41 a) successfully locates the global minimum, but QNA-XEL with $n < 1$ have 15 to 17 paths that do so. The high number of paths that reach the global minimum is due to the simplicity of the energy landscape. Some QNA-XEL paths never reach the minimum because an inaccurate estimated modified Hessian matrix gives a step almost perpendicular to the gradient that yields a very small step size. As the result the path never reaches the minimum before the simulation reaches the maximum iteration. On the complex energy landscape (Figure 5.41 b) far fewer paths from all algorithms can locate the global minimum. In each algorithm only a few values of $n \neq 1$ have more successful paths than $n = 1$. In QNA-XEL $n < 1$ cases have more successful paths than $n > 1$ but in BFGS-XEL $n > 1$ cases have more successful paths than $n \leq 1$ and in Newton-XEL all cases have about the same number of paths. As $n < 1$ QNA-XEL has a larger step size and while $n < 1$ cases have more successful paths than $n > 1$ QNA-XEL with $n = 0.125$ have fewer successful paths than $n = 0.5$ or 0.25. This suggests that a larger step size gives a more favorable condition for a global minimum search but a step size too large can cause $\hat{K}^*$ to diverge.

With a heuristic exhaustive line search, the results of the Newton-XEL (*Hessian only*), the QNA-XEL (*Hessian only*), and the BFGS-XEL (*Hessian only*) on the simple

energy landscape (Figure 5.42 a) are the same as those of the Newton-XEL, the QNA-XEL, and the BFGS-XEL. On the complex energy landscape (Figure 5.42 b) far fewer paths from all algorithms can locate the global minimum. For all algorithms $n < 1$ cases have more successful paths than $n \geq 1$ cases.

Results are inconclusive as to whether the XEL has positive or negative impact on the global search. The XEL does not change the performance in a global search for the Newton-XEL, the Newton-XEL (*Hessian only*), the BFGS-XEL, and the BFGS-XEL (*Hessian only*) on the simple energy landscape. The XEL may reduce or improve the performance on the complex energy landscape, but more often it reduces. The XEL with $n < 1$ reduces the performance in a global search in the QNA-XEL and the QNA-XEL (*Hessian only*) except when they are used with the heuristic exhaustive line search on the complex energy landscape. With the bracketing and Brent line search the performance is slightly varied by the values of n but with the heuristic exhaustive line search the variation can be significant. The XEL impact on the global search performance is further investigated in the optimization with a probabilistic search presented in Section 5.2.

5.1.4 Conclusion of XEL Algorithms without Probabilistic Search

To study the effect of the XEL on Newton's method, the QNA, and the BFGS, the Newton-XEL, the Newton-XEL (*Hessian only*), the QNA-XEL, the QNA-XEL (*Hessian only*), the BFGS-XEL, and the BFGS-XEL (*Hessian only*) are implemented without a probabilistic search on a simulated two-dimensional energy landscape with a bracketing and Brent or a heuristic exhaustive line search. The results are evaluated in terms of quality and speed by comparing the converged energy and the number of iterations. The paths generated by the investigated algorithms are compared and the the impact of the XEL on the global search is investigated.

A path comparison shows that paths and step sizes are dependent on n but this

dependency can be reduced by the shape of the energy landscape or a line search algorithm. Paths tend to have longer steps when $n < 1$ but shorter steps when $n > 1$ which is in agreement with Guideline *ii.* in Table 4.1. Some QNA-XEL or BFGS-XEL paths show that poor Hessian matrix estimation can result in poor speed and even divergence while $n > 1$ results in a more accurate estimation. This is in agreement with Guideline *iv.* and Guideline *iii.* since increasing n results in better convergence and small steps which implies a bounded magnitude of λ.

The percentages of paths that have lower, equal, or higher energy or number of iterations compared to $n = 1$ show that $n > 1$ yields better quality on the simple energy landscape but $n < 1$ yields better quality on the complex energy landscape. Since $n > 1$ gives a more accurate Hessian matrix estimation, paths can converge quickly to a minimum. This works well on the simple energy landscape with only one minimum but on the complex energy landscape a local minimum nearest to the starting point may not have the low energy and therefore $n > 1$ may not result in a lower energy and $n < 1$ with longer steps tends to converge a lower minimum energy by covering more distance. This suggests that $n < 1$ should be used at the beginning of optimization for farther exploration and $n > 1$ should be used at the end for convergence. This is in agreement with Guideline *v.* in Table 4.1.

On average the bracketing and Brent line search yields better quality and speed than the heuristic exhaustive line search and thus it is implemented in the optimization with a probabilistic search presented next in Section 5.2. The algorithms investigated in the next section are the QNA-XEL, the QNA-XEL (*Hessian only*), the BFGS-XEL, and the BFGS-XEL (*Hessian only*). Results in this section show that with the bracketing and Brent line search the QNA-XEL and the QNA-XEL (*Hessian only*) yield better quality and speed when $n > 1$, the BFGS-XEL yields the same quality with different values of n but slightly higher speed when $n > 1$, and the BFGS-XEL (*Hessian only*) yields better quality when $n < 1$ and slightly higher

speed when $n > 1$. These will be compared with those in Section 5.2.

Lastly, results show that the XEL impact on the global search is inconclusive. This impact is further investigated in Section 5.2

5.2 XEL Algorithms with Probabilistic Search

This section investigates the effects of XEL on the optimization algorithm combined with a probabilistic search. The QNA-XEL, the QNA-XEL (*Hessian only*), the BFGS-XEL, and the BFGS-XEL (*Hessian only*) are implemented with the bracketing and Brent line search on the protein conformation prediction software Rosetta. The Newton-XEL and the Newton-XEL (*Hessian only*) is not studied because the second order derivatives are not be available in this platform.

5.2.1 Rosetta

Using both analytical and probabilistic methods, Rosetta is a computer program that predicts protein conformation as it constructs decoys or initial configurations from pre-generated fragments (decoy generation) and refines them by perturbing and then minimizing the predicted configurations (refinement protocol) [11] which is called the Monte Carlo-minimization approach.

Decoy generation consists of three components: fragment generation, scoring functions, and fragment assembly. A decoy is generated during the fragment assembly by piecing the generated fragments together which is guided by the scoring function. Fragments are the structural templates of small windows of a protein sequence and potential shapes that can be assumed by corresponding windows of the protein molecule. Windows are defined as 3- or 9-residue sections of a protein sequence. By comparing the protein sequence to the protein database, fragments are generated for all possible windows in the protein sequence, which are then ranked according to sequence and secondary structure similarity.

The second component essential for both decoy generation and refinement protocol are the scoring functions. Two types of scoring functions, centroid and full atom, are available for course-grained and atomic-level descriptions of the molecule structure. Despite their differences, both include energy functions, such as van der Waals

interactions and electrostatics, and non-energy potentials, such as the Ramachandran torsion preferences [10].

Lastly, during fragment assembly a decoy is constructed as the parts of the molecule corresponding to randomly chosen sequence windows assume the shapes of fragments randomly chosen from high-ranking fragments. After each fragment is inserted, the scoring functions of the entire protein are evaluated, and using the Monte Carlo method this insertion is kept or discarded. At the beginning of the fragment assembly fewer scoring functions are strategically evaluated to encourage local folding of the molecule and toward the end of the process all scoring functions are evaluated to encourage compact packing.

After the decoys are generated, a refinement protocol perturbs each decoy configuration with several small moves and performs a minimization after each perturbation using an analytical optimization algorithm. Here, the QNA-XEL, the QNA-XEL (*Hessian only*), the BFGS-XEL, and the BFGS-XEL (*Hessian only*) along with the bracketing and Brent line search are implemented in the centroid refinement protocol and compared to the QNA and the BFGS algorithm. After each perturbation and each minimization, the Monte Carlo method is used to determine if the new configuration should be accepted or rejected. If accepted, the next set of the perturbation and minimization is done on the new configuration. Otherwise, the next set is done on the previous configuration. The centroid refinement protocol which uses score functions describing scores between residues is chosen to show proof-of-concept because it is less computational expensive than the full atom refinement protocol which is used in [11].

Shown as a flowchart in Figure 5.43, the Rosetta centroid refinement protocol repeats the following two steps six times: 1) minimize the energy on the entire chain, 2) perturb and minimize the energy on portions of the chain several times depending on the chain length. The minimization in both 1) and 2) uses the QNA-XEL, the

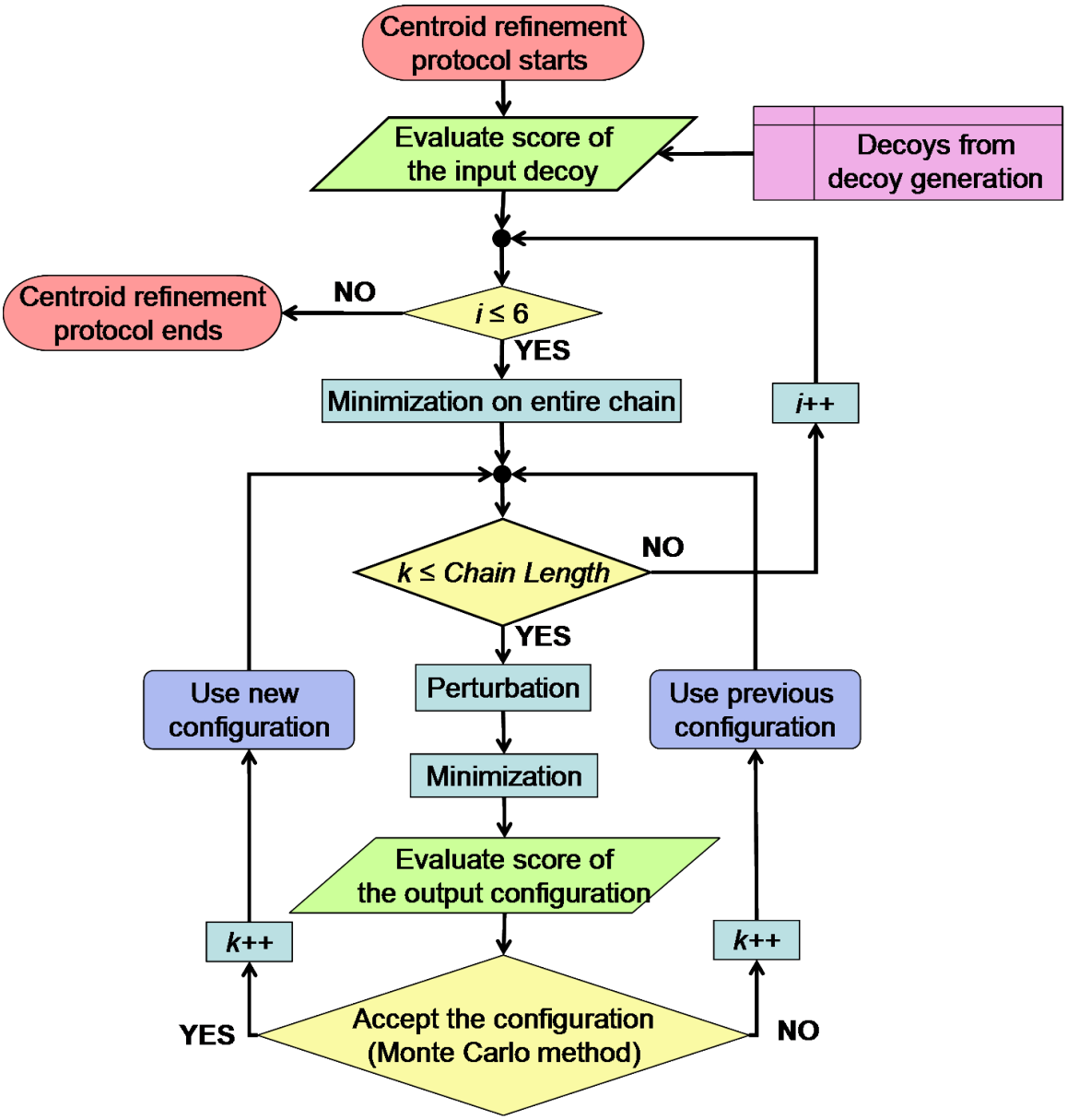

Figure 5.43: Rosetta centroid refinement protocol. The QNA-XEL, the QNA-XEL (*Hessian only*), the BFGS-XEL, and the BFGS-XEL (*Hessian only*) are implemented during the minimization process.

QNA-XEL (*Hessian only*), the BFGS-XEL, and the BFGS-XEL (*Hessian only*) to locate the configuration with the minimum score, which is the weighted sum of the scoring functions and implies an energy function E. Each minimization converges and stops when the change in the score is less than 0.5% or when iterations reach a maximum of 200.

Simulations are performed on several proteins, each analyzed with several values of n including $n = 1$ which corresponds to the unmodified energy landscape. Proteins used in the simulation are those provided in the Rosetta software package and the Robetta server [37], which contains fragment files and other necessary files to generate decoys, some of which are not in the protein database. Five small proteins analyzed here are 1ubq, 1d3z, dom6, hetr, and mdmi, which have 76, 82, 126, 149, and 304 residues, respectively. Various sizes are chosen to determine if the size of the protein effects the results and the range is limited to the size of an average protein because of the available computational resources.

For each protein 50 different decoys are generated and used as starting configurations in the refinement for each n value. Because of the probabilistic nature of the methods used in decoy generation and the perturbation in the refinement protocol, the number of decoys is chosen so that the sample size is sufficient to produce results that represent the performance of the optimization algorithm while small enough for the available computational resources. Simulations on 50 decoys for each n case take 3:40 to 32:18 hours depending on the protein on an AMD Athlon^{TM} XP 3200+ (2.1 GHz) or AMD Athlon^{TM} 64 Processor 3500+ (2.2 GHz) computers. Typical runtime for an accurate prediction of the conformation of a protein is 150 days.

The results are evaluated in terms of quality, speed, and efficiency. For quality an average score improvement, a lowest score, and similarity to the native conformation are determined from output scores and configurations. These quantities are defined and discussed in Sections 5.2.2, 5.2.3, and 5.2.6 respectively. The similarity of output

configurations to the native conformation are evaluated in terms of TM score [88], GDT score [79, 80], MaxSub score [62], and RMSD [32, 33]. These quantities are discussed in Section 5.2.6. For speed an average iteration is determined from the number of total iterations. This quantity is defined and discussed in Section 5.2.4. Efficiency is determined from the average score improvement and the average iteration. This quantity is defined and discussed in Section 5.2.5. Section 5.2.7 discusses results from a further analysis studying the effect of XEL when its simulations are ran with the same number of iterations as that of the unmodified case.

5.2.2 Score Improvement

This section gives the evaluation of the QNA-XEL, the QNA-XEL (*Hessian only*), the BFGS-XEL, and the BFGS-XEL (*Hessian only*) on quality by determining the score improvement quantities described below.

After the refinement process, the scores E of the obtained configurations are evaluated. The average of score improvement $AveSI$ is defined as the average of the score improvements SI which are the differences between scores of the resulting configurations (outputs) and those of the decoys (inputs):

$$AveSI = \frac{1}{N}\sum_{i=1}^{N} SI_i \tag{5.2}$$

$$SI_i = E_{m,i} - E_{0,i}$$

where N is the number of decoys ($N = 50$), SI_i is the score improvement of the i^{th} decoy, $E_{m,i}$ is the output score or the score at the m^{th} or last iteration for the i^{th} decoy, and $E_{0,i}$ is the input score or the score at the 0^{th} iteration for the i^{th} decoy. Since the refinement protocol is a minimization process, the score of a resulting configuration is considered "improved" when its value is less than that of the input decoy. Therefore, the lower SI and $AveSI$ values imply better results.

The results from different cases of n are compared to those from unmodified cases

$(n = 1)$ by subtracting the $AveSI$ values of unmodified cases from those of the XEL cases:

$$\%\Delta AveSI_n = \frac{AveSI_n - AveSI_{n=1}}{|AveSI_{n=1}|} \times 100 \tag{5.3}$$

where $\%\Delta AveSI_n$ is the percent difference of the average score improvement compared to the unmodified case, $AveSI_n$ is the $AveSI$ value of the XEL case, and $AveSI_{n=1}$ is the $AveSI$ value of unmodified case. The lower $\%\Delta AveSI_n$ indicates that the XEL case yields greater improvement and therefore a better result.

5.2.2.1 QNA-XEL and QNA-XEL (Hessian only)

Figures 5.44 and 5.45 display $AveSI$ versus various values of n for each protein for the QNA-XEL and the QNA-XEL (*Hessian only*), respectively. The black dashed horizontal lines indicate the $AveSI$ from the $n = 1$ cases and the red dotted lines above and below the black dashed lines indicate $\pm 5\%$ deviations. Some red dotted lines may not be visible if all data lie inside the upper or lower boundary. Some black dashed lines may not be visible if the data points for $n = 1$ are at the lower ends of the data ranges. Note that lines connecting data points in all figures presented here by no means imply any relationships nor trends, but they are only used for clarity. The values of n are shown on the x-axes which are neither a linear nor logarithmic scale. The range of n values is selectively investigated because it sufficiently represents the performance of the algorithms. Also, as n becomes lower than 2^{-9} the performance in terms of the score improvement reduces and as n becomes higher than 10, the optimization algorithm diverges.

For the QNA-XEL Figure 5.44 shows that all test proteins have the lowest $AveSI$ when $n = 1$, 2, or 6 and that all $n > 1$ cases yield better results than those of the $n < 1$ cases. While the unmodified case $(n = 1)$ yields the best or the second best results in all proteins, most $n > 1$ cases are within the $\pm 5\%$ deviations which implies similar performance. For the QNA-XEL (*Hessian only*) Figure 5.45 shows that the

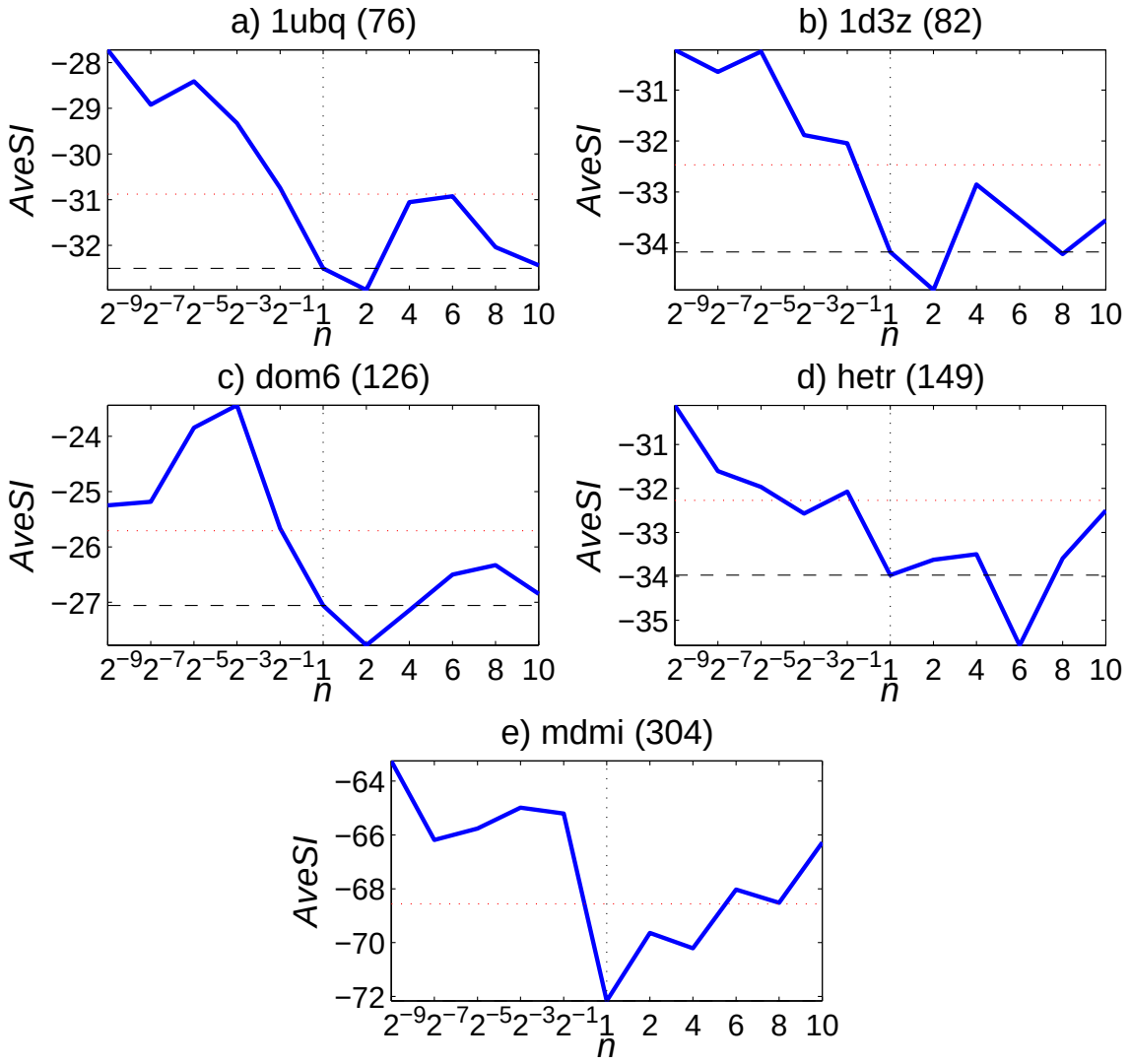

Figure 5.44: QNA-XEL: The average of the score improvement ($AveSI$) versus various values of n in a) 1d3z, b) 1ubq, c) dom6, d) hetr, and e) mdmi proteins. Lower values indicate greater improvements. The red dotted lines above and below black dashed horizontal lines indicate 5% deviations above and below the $AveSI$ of the $n = 1$ cases.

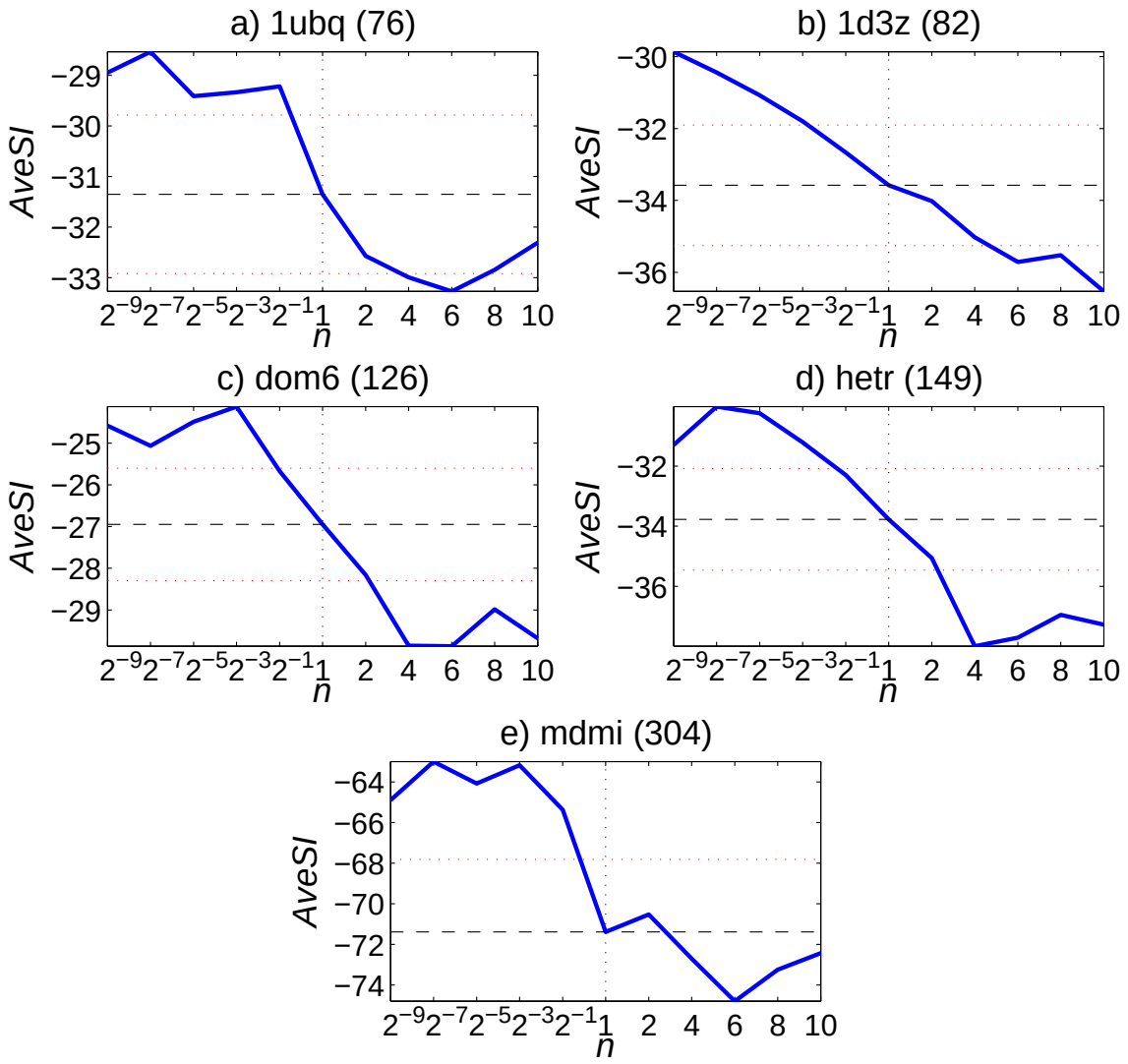

Figure 5.45: QNA-XEL (*Hessian only*): The average of the score improvement (*AveSI*) versus various values of n in a) 1d3z, b) 1ubq, c) dom6, d) hetr, and e) mdmi proteins. Lower values indicate greater improvements. The red dotted lines above and below black dashed horizontal lines indicate 5% deviations above and below the *AveSI* of the $n = 1$ cases.

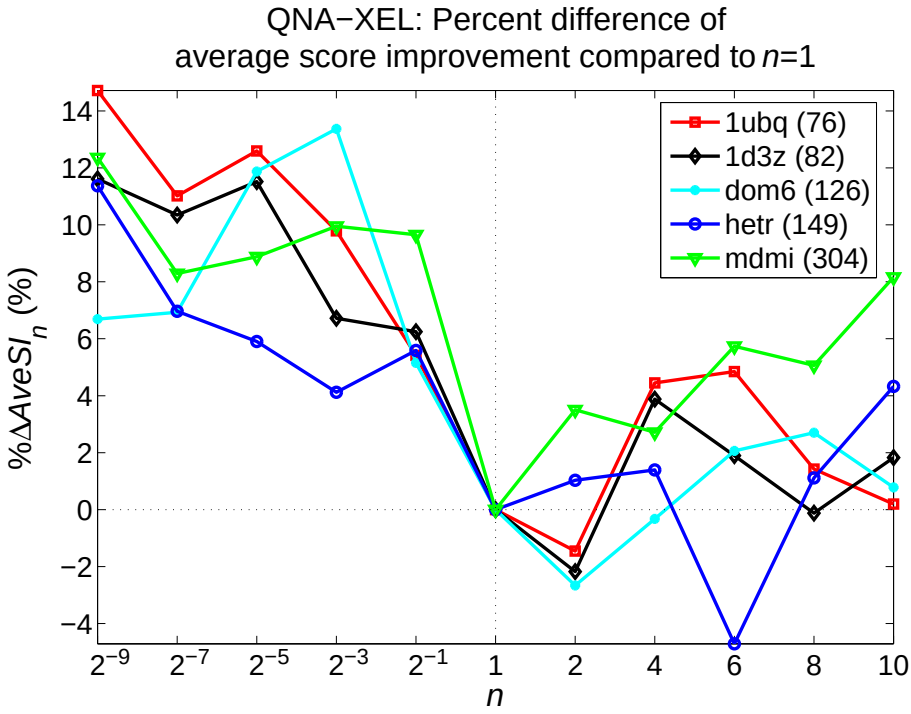

Figure 5.46: QNA-XEL: The percent difference between the average score improvement of the various values of n and those of unmodified energy landscape ($n = 1$) case, $\Delta AveSI_n$. Negative values indicate that the XEL yields better results than the unmodified case. Lower values indicate greater improvements.

test proteins have the best results when $n = 4$, 6, or 10 and all but one $n > 1$ cases yield better results than those of $n < 1$ and $n = 1$ cases. In fact, proteins 1ubq, 1d3z, dom6, and hetr have more than 5% lower $AveSI$ than $n = 1$.

Note that when $n = 1$ both the QNA-XEL and the QNA-XEL (*Hessian only*) become the QNA and therefore they should yield the same results. However, results may be slightly different since the results of $n = 1$ shown in each algorithm are from different runs and there is the probabilistic nature of the methods used in the perturbation in the refinement protocol. This also applies to the BFGS-XEL and the BFGS-XEL (*Hessian only*).

Figures 5.46 and 5.47 display the percent difference of the average score improvement compared to the unmodified case $\%\Delta AveSI_n$ for the QNA-XEL and the QNA-XEL (*Hessian only*), respectively. Negative values indicate that the XEL yields better

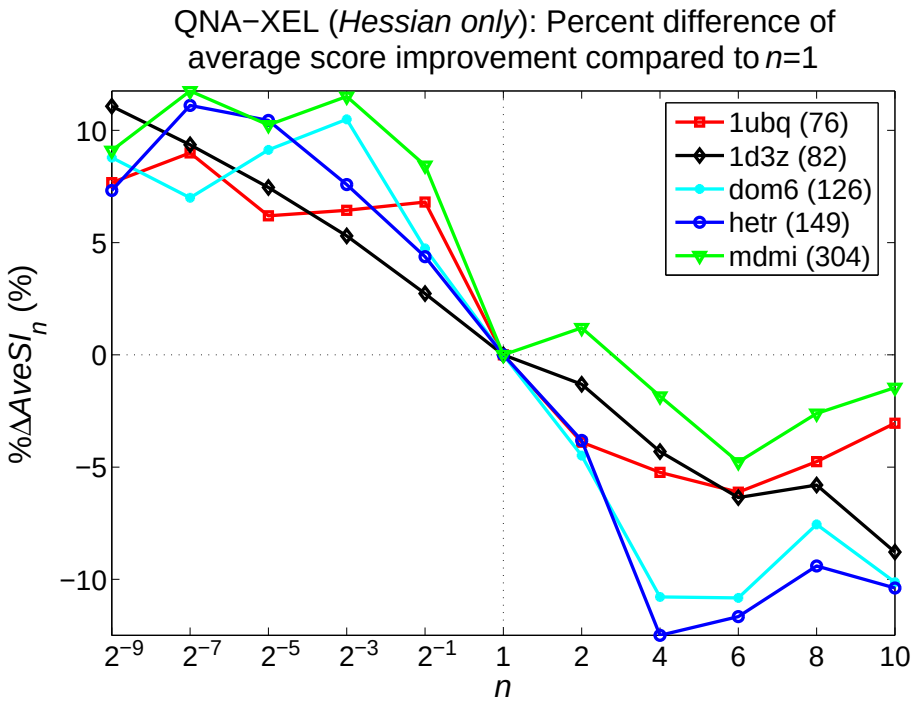

Figure 5.47: QNA-XEL (*Hessian only*): The percent difference between the average score improvement of the various values of n and those of unmodified energy landscape ($n = 1$) case, $\Delta AveSI_n$ versus various values of n. Negative values indicate that the XEL yields better results than the unmodified case. Lower values indicate greater improvements.

results than the unmodified case. Lower values indicate greater score improvements. For the QNA-XEL Figure 5.46 illustrates consistent results throughout all test proteins as $n < 1$ yields worse results than $n \geq 1$ and $n > 1$ yields results within $\pm 5\%$ of those from $n = 1$. No correlation between the protein size and the value of $\%\Delta AveSI_n$ can be established. For the QNA-XEL (*Hessian only*) Figure 5.47 illustrates consistent results throughout all test proteins which are similar to those from the QNA-XEL but $n > 1$ generally yields lower results than $n = 1$ and in some cases more than 10% lower. For $n > 2$ the value of $\%\Delta AveSI_n$ seems dependent on the protein size as larger proteins yield lower $\%\Delta AveSI_n$ except for the protein mdmi.

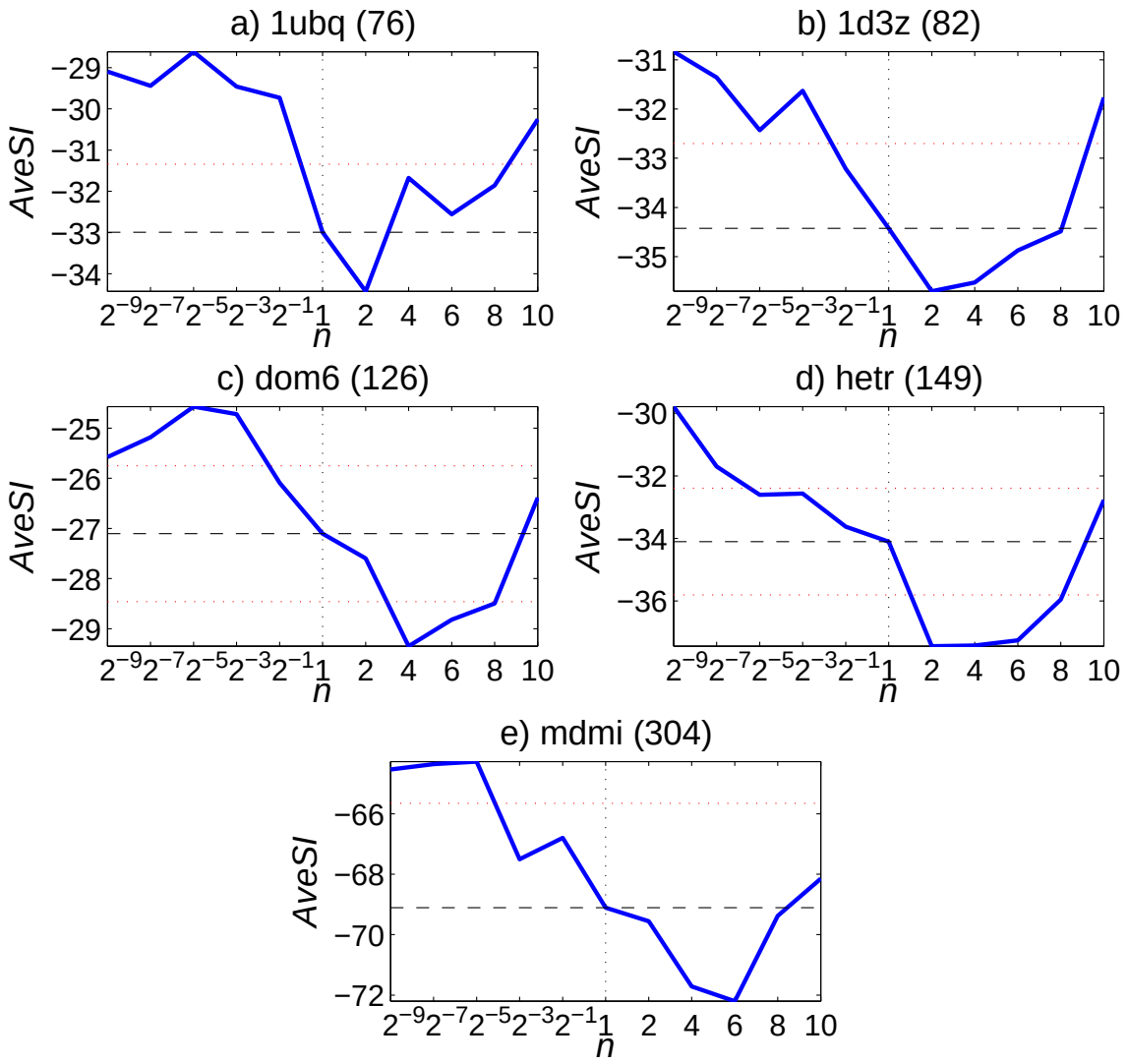

Figure 5.48: BFGS-XEL: The average of the score improvement ($AveSI$) versus various values of n in a) 1d3z, b) 1ubq, c) dom6, d) hetr, and e) mdmi proteins. Lower values indicate greater improvements. The red dotted lines above and below black dashed horizontal lines indicate 5% deviations above and below the $AveSI$ of the $n = 1$ cases.

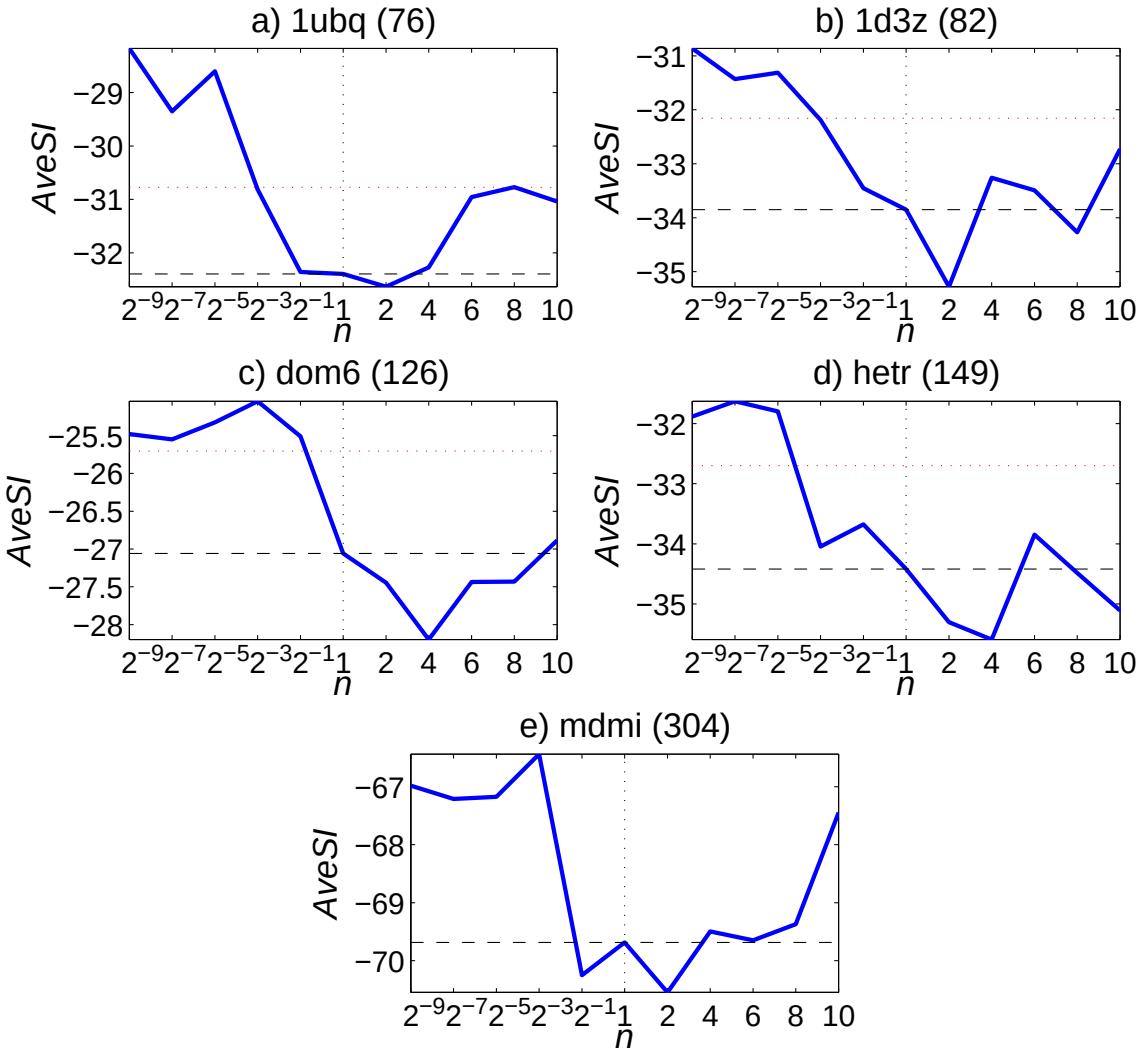

Figure 5.49: BFGS-XEL (*Hessian only*): The average of the score improvement ($AveSI$) versus various values of n in a) 1d3z, b) 1ubq, c) dom6, d) hetr, and e) mdmi proteins. Lower values indicate greater improvements. The red dotted lines above and below black dashed horizontal lines indicate 5% deviations above and below the $AveSI$ of the $n = 1$ cases.

5.2.2.2 BFGS-XEL and BFGS-XEL (Hessian only)

Figures 5.48 and 5.49 display $AveSI$ versus various values of n for each protein for the BFGS-XEL and the BFGS-XEL (*Hessian only*), respectively. The black dashed horizontal lines indicate the $AveSI$ from the $n = 1$ cases and the red dotted lines above and below the black dashed lines indicate $\pm 5\%$ deviations. Some red dotted lines may not be visible if all data lie inside the upper or lower boundary. Lower values indicate greater improvement.

For the BFGS-XEL Figure 5.48 shows that all test proteins have the lowest $AveSI$ when $n = 2$, 4, or 6 and that most $n > 1$ cases yield better results than those of the $n < 1$ cases and lie within $\pm 5\%$ boundaries of $n = 1$ which implies similar performance. For the BFGS-XEL (*Hessian only*) Figure 5.49 shows that the test proteins have the best results when $n = 2$ or 4 and most $n < 1$ cases yield worse results than those of $n \geq 1$. In fact, all but one $n < 1$ case yields worse results than those of the $n = 1$ cases.

Figures 5.50 and 5.51 display the percent difference of the average score improvement compared to the unmodified case $\%\Delta AveSI_n$ for the BFGS-XEL and the BFGS-XEL (*Hessian only*), respectively. For the BFGS-XEL Figure 5.50 illustrates consistent results throughout all test proteins as $n < 1$ yields worse results than $n \geq 1$ and most $n > 1$ yields results within $\pm 5\%$ or lower than those from $n = 1$. For $n > 2$ the value of $\%\Delta AveSI_n$ tends to be dependent on the protein size as larger proteins yield lower $\%\Delta AveSI_n$ except for protein mdmi. For the BFGS-XEL (*Hessian only*) Figure 5.51 illustrates consistent results throughout all test proteins which are similar to those from the BFGS-XEL but all $n > 1$ cases yields results within $\pm 5\%$ of those from $n = 1$. No correlation between the protein size and the value of $\%\Delta AveSI_n$ can be established.

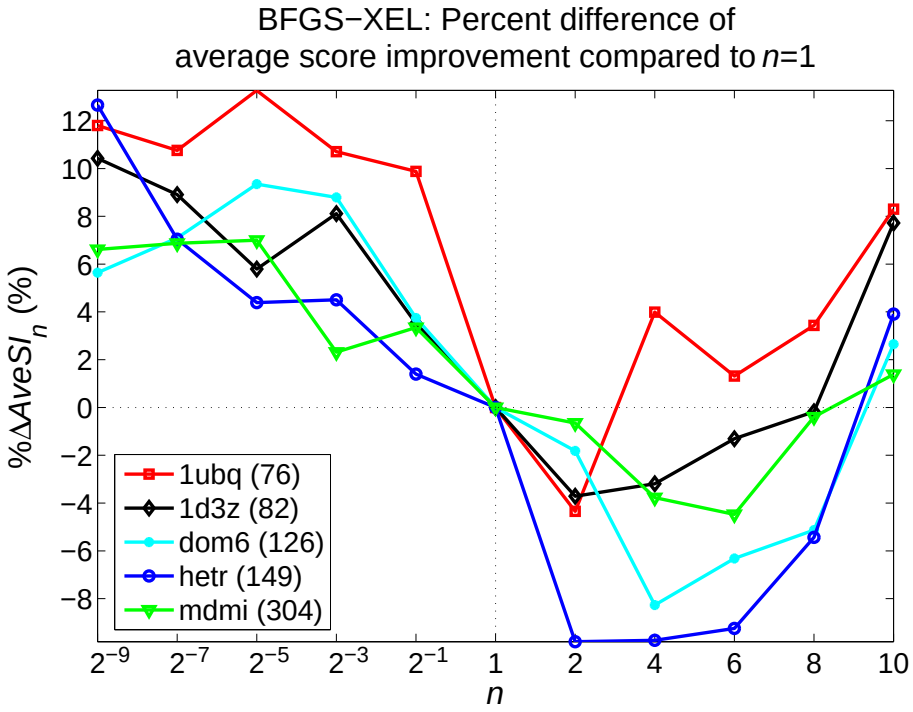

Figure 5.50: BFGS-XEL: The percent difference between the average score improvement of the various values of n and those of unmodified energy landscape ($n = 1$) case, $\Delta AveSI_n$. Negative values indicate that the XEL yields better results than the unmodified case. Lower values indicate greater improvements.

5.2.2.3 Conclusion of score improvement

The *AveSI* from configurations generated by the QNA-XEL, the QNA-XEL (*Hessian only*), the BFGS-XEL, and the BFGS-XEL (*Hessian only*) with different values of n in proteins 1ubq, 1d3z, dom6, hetr, and mdmi are compared. Results show that for all algorithms $n > 1$ yields better quality than $n < 1$ and in many cases yield better quality than $n = 1$. The same conclusion can also be drawn from results in Figures A.2 through A.8 a) through c), g), and h) in Appendix A which present the percentage of decoys with lower, equal, or higher score than those of $n = 1$ for all test algorithms. This is also in agreement with a conclusion from Section 5.1 and Guideline *iv.* in Table 4.1.

Note that although results in this section show that $n < 1$ yields worse quality than $n = 1$, the worst case has only 15% higher *AveSI*.

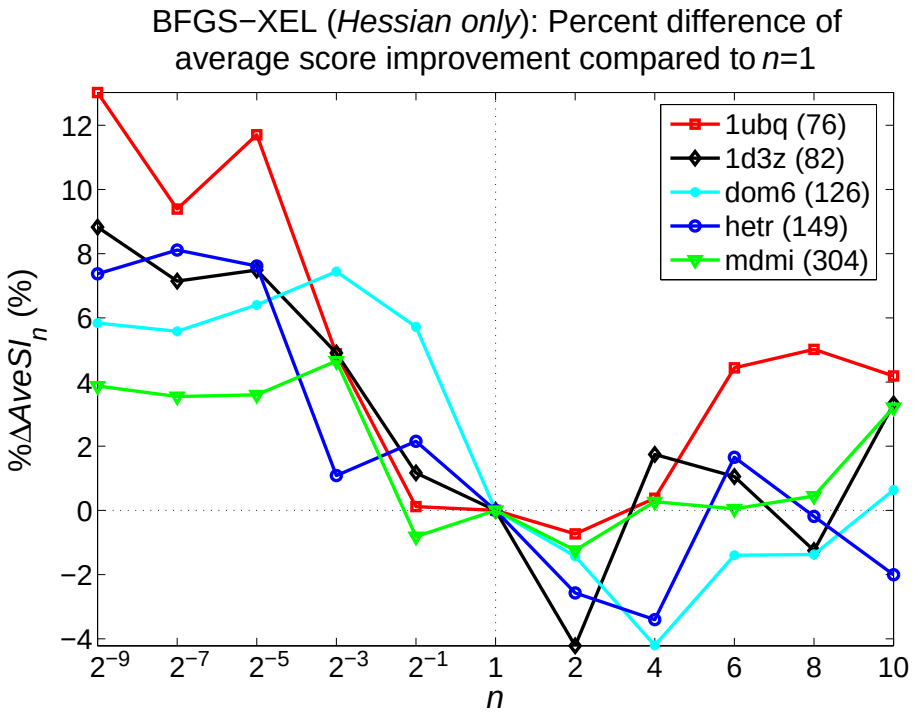

Figure 5.51: BFGS-XEL (*Hessian only*): The percent difference between the average score improvement of the various values of n and those of unmodified energy landscape ($n = 1$) case, $\Delta AveSI_n$. Negative values indicate that the XEL yields better results than the unmodified case. Lower values indicate greater improvements.

5.2.3 Lowest Score

In addition to performance in terms of quality, the lowest score indicates performance in terms of global search performance or ability to find the global solution. Since results can not be directly compared to the native state or the global solution, the lowest score E_{lowest} among the final scores of all decoys are evaluated and compared. Additionally, the percent difference of the lowest score $\%\Delta E_{\text{lowest}}$ compared to $n = 1$ is determined which is defined as

$$\%\Delta E_{\text{lowest}} = \frac{(E_{\text{lowest}})_n - (E_{\text{lowest}})_{n=1}}{|(E_{\text{lowest}})_{n=1}|} \times 100 \tag{5.4}$$

where $(E_{\text{lowest}})_n$ is the lowest score of the XEL case and $(E_{\text{lowest}})_{n=1}$ is the lowest score of the unmodified case. Lower values of E_{lowest} and $\%\Delta E_{\text{lowest}}$ imply better quality and global search performance.

5.2.3.1 QNA-XEL and QNA-XEL (*Hessian only*)

The lowest scores from different values of n in proteins 1ubq, 1d3z, dom6, hetr, and mdmi are compared in Figures 5.52 and 5.53 for the QNA-XEL and the QNA-XEL (*Hessian only*), respectively. The red dotted lines indicate ±5% deviations of the lowest scores from the $n = 1$ cases indicated by black dashed horizontal lines. Some red dotted lines may not be visible if all data lie inside the upper or lower boundary. Some black dashed lines may not be visible if the data points for $n = 1$ are at the lower ends of the data ranges.

For the QNA-XEL Figure 5.52 shows that three of five test proteins have the lowest score when $n = 10$ and the others have the lowest score when $n = 2^{-3}$ or 10. The deviation of the lowest score between different values of n is higher than that of the $AveSI$ but on average in most proteins $n > 1$ has lower lowest score than $n < 1$ which agrees with results presented in the previous section. For the QNA-XEL (*Hessian only*) Figure 5.53 shows that two test proteins have the best results when $n = 1$ and the others have the lowest score when $n = 2^{-7}$, 6, or 10. Again, the deviation of the lowest score between different values of n is higher than that of the $AveSI$ but on average in most protein $n > 1$ has lower lowest score than $n < 1$.

Figures 5.54 and 5.55 display the percent difference of the lowest score $\%\Delta E_{\text{lowest}}$ compared to $n = 1$ for the QNA-XEL and the QNA-XEL (*Hessian only*), respectively. Negative values indicate that the XEL yields better results than the unmodified case. Lower values indicate better performance. For the QNA-XEL Figure 5.54 illustrates that in all test proteins $n \neq 1$ yields worse results than $n = 1$ on average but three cases yield significantly better results than others. In protein hetr when $n = 6$, 8, or 10 the lowest score can be more than 20% lower than that of $n = 1$. The range of $\%\Delta E_{\text{lowest}}$ is between -31% to 28%, most values fall within ±15% range. No correlation between the protein size and the value of $\%\Delta AveSI_n$ can be established. For the QNA-XEL (*Hessian only*) Figure 5.55 illustrates that in all test proteins

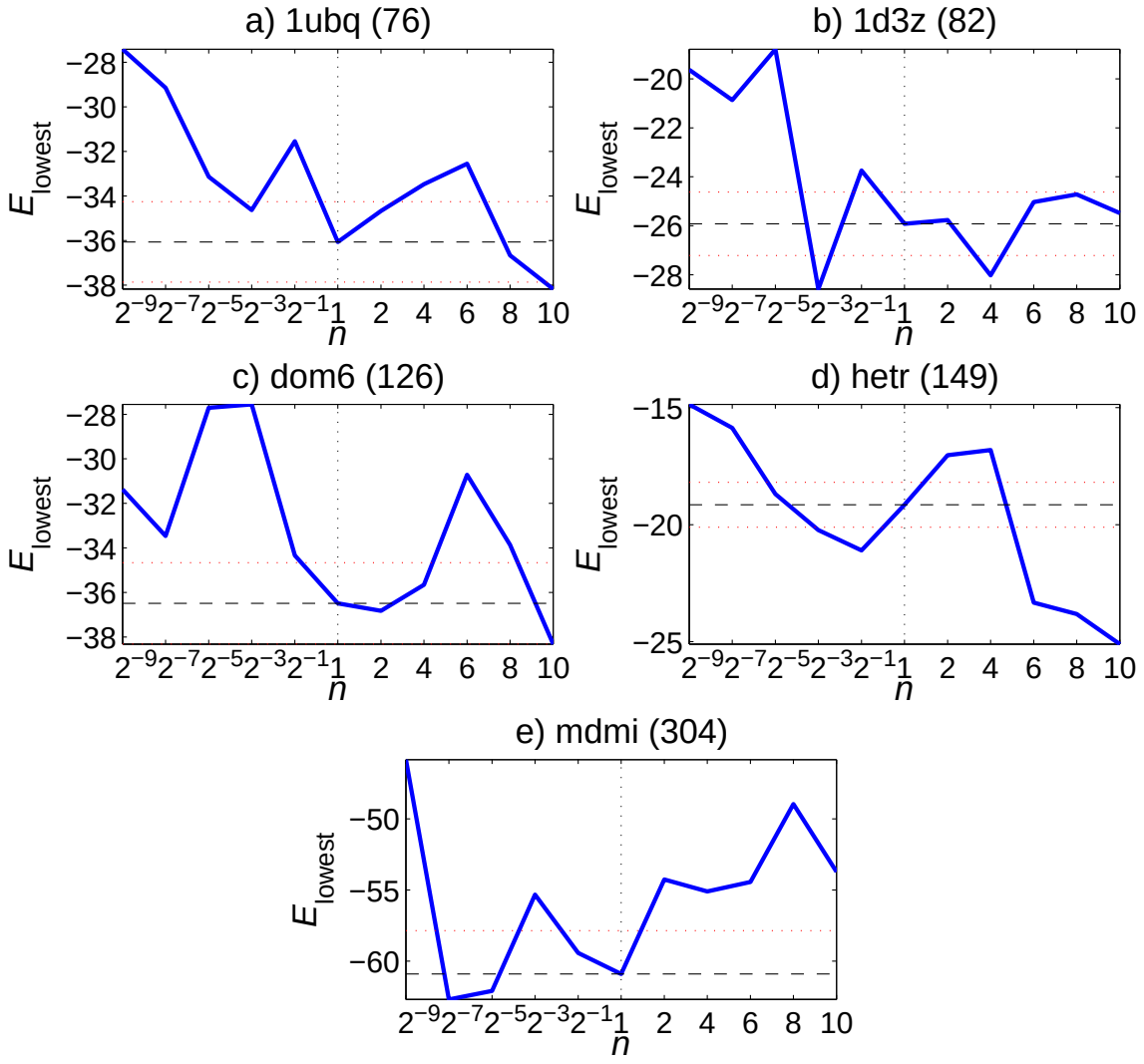

Figure 5.52: QNA-XEL: The lowest scores (E_{lowest}) versus various values of n in a) 1ubq, b) 1d3z, c) dom6, d) hetr, and e) mdmi. Lower values indicate better performance. The red dotted lines above and below black dashed horizontal lines indicate 5% deviations above and below the lowest scores from the $n = 1$ cases.

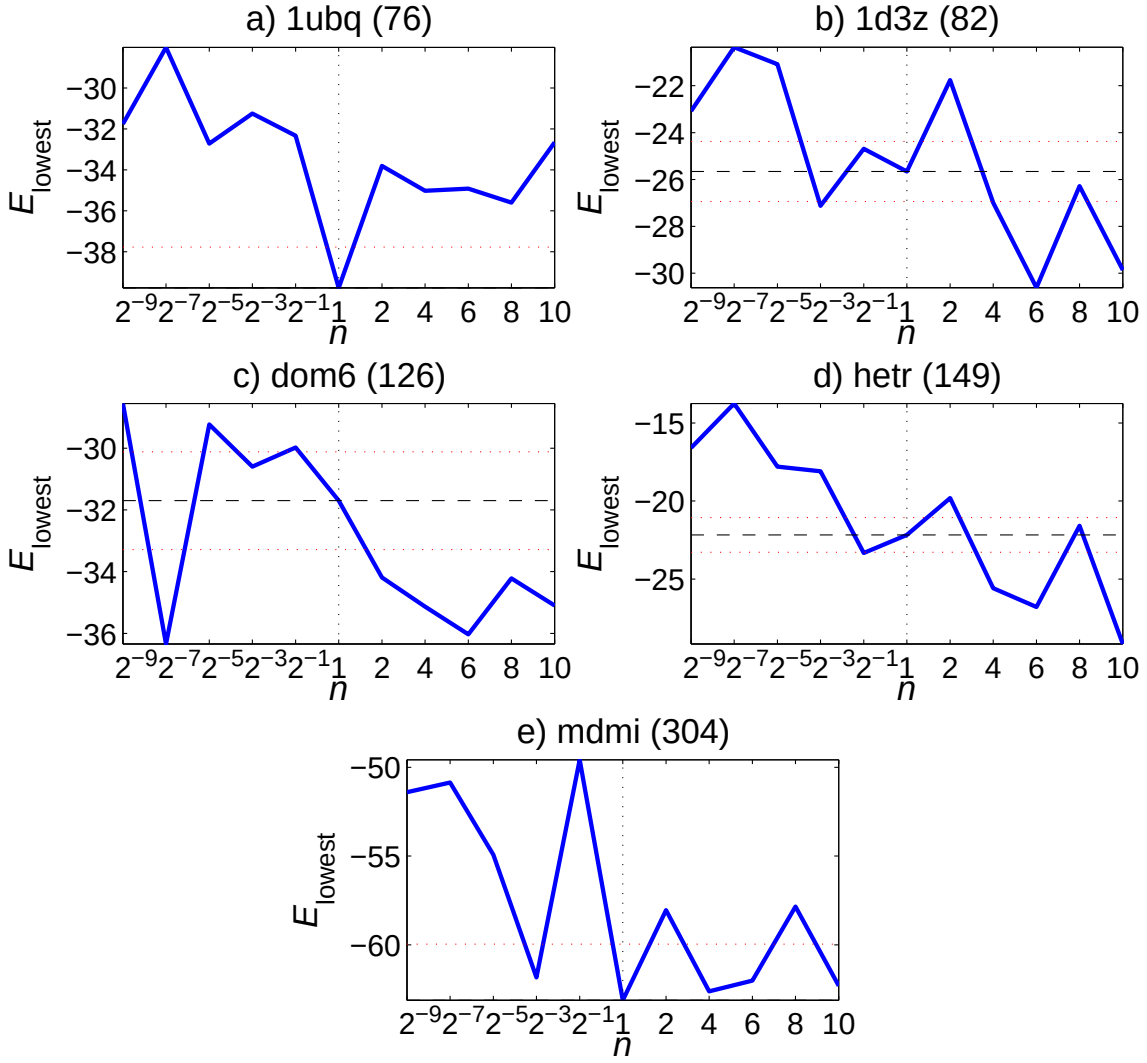

Figure 5.53: QNA-XEL (*Hessian only*): The lowest scores (E_{lowest}) versus various values of n in a) 1ubq, b) 1d3z, c) dom6, d) hetr, and e) mdmi. Lower values indicate better performance. The red dotted lines above and below black dashed horizontal lines indicate 5% deviations above and below the lowest scores from the $n = 1$ cases.

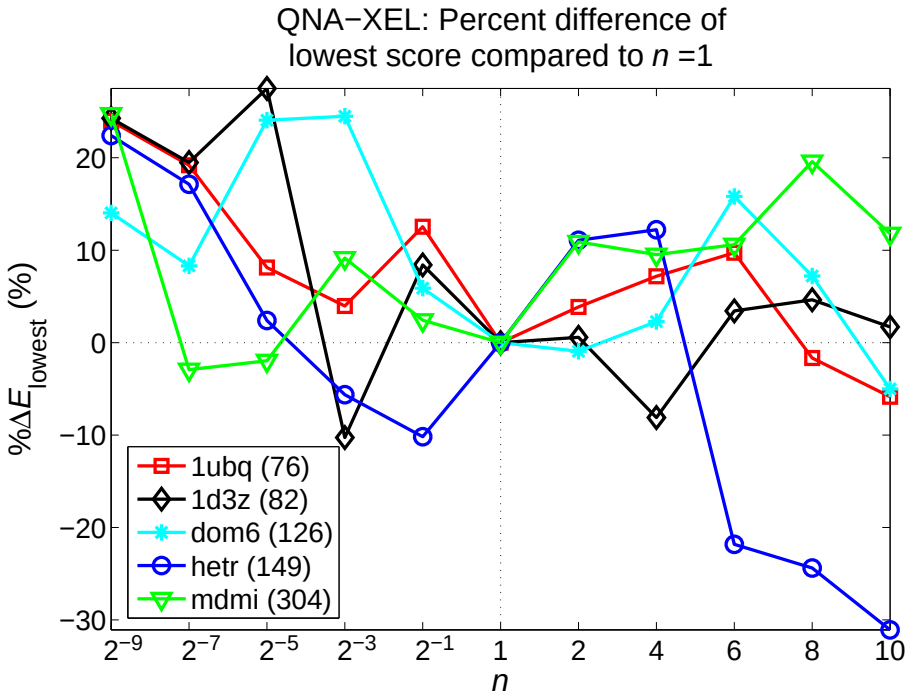

Figure 5.54: QNA-XEL: The percent difference of the lowest scores ($\%\Delta E_{\text{lowest}}$) compared to $n = 1$ in a) 1ubq, b) 1d3z, c) dom6, d) hetr, and e) mdmi. Lower values indicate better performance.

$n \neq 1$ yields worse results than $n = 1$ on average. The worst cases of the QNA-XEL (*Hessian only*) have worse results than those in the QNA-XEL. The range of $\%\Delta E_{\text{lowest}}$ is between -31% to 88%, most values fall within ±20% range. No correlation between the protein size and the value of $\%\Delta E_{\text{lowest}}$ can be established.

5.2.3.2 BFGS-XEL and BFGS-XEL (Hessian only)

The lowest scores from different values of n in proteins 1ubq, 1d3z, dom6, hetr, and mdmi are compared in Figures 5.56 and 5.57 for the BFGS-XEL and the BFGS-XEL (*Hessian only*), respectively. The red dotted lines indicate ±5% deviations of the lowest scores from the $n = 1$ cases indicated by black dashed horizontal lines. Some red dotted lines may not be visible if all data lie inside the upper or lower boundary.

For the BFGS-XEL Figure 5.56 shows that three of five test proteins have the lowest score when $n = 4$ and the others have the lowest score when $n = 2$ or 8. The

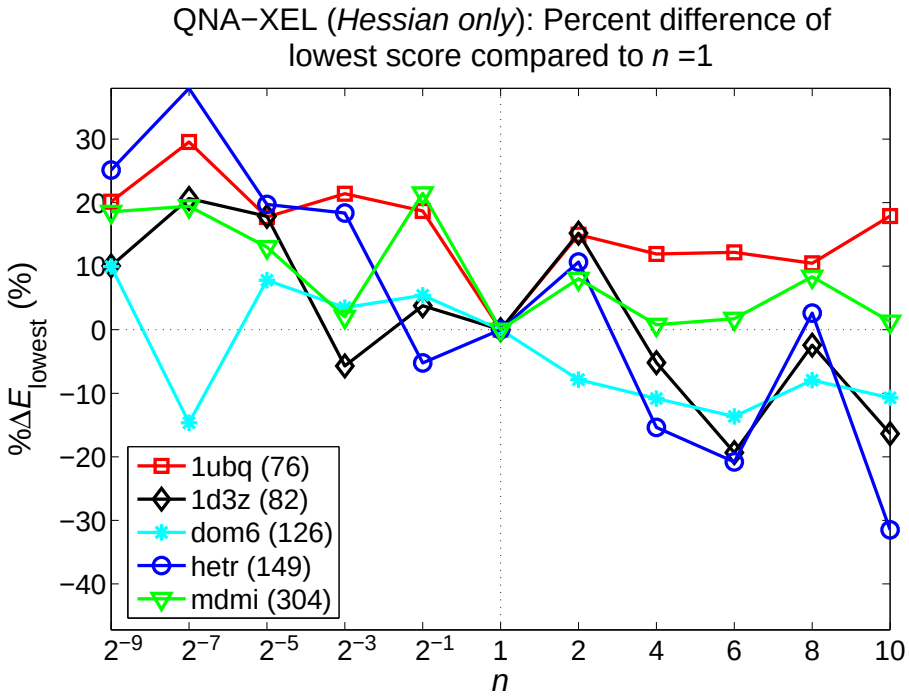

Figure 5.55: QNA-XEL (*Hessian only*): The percent difference of the lowest scores ($\%\Delta E_{\text{lowest}}$) compared to $n = 1$ in a) 1ubq, b) 1d3z, c) dom6, d) hetr, and e) mdmi. Lower values indicate better performance.

deviation of the lowest score between different values of n is higher than that of the *AveSI* but on average in most proteins $n > 1$ yield slightly lower lowest score than $n < 1$ which agrees with results presented in the previous section. For the BFGS-XEL (*Hessian only*) Figure 5.57 shows that each test protein has the lowest score when $n \neq 1$. Again, the deviation of the lowest score between different values of n is higher than that of the *AveSI* but for proteins 1d3z, hetr, and mdmi, most lowest scores are lower or inside the boundaries.

Figures 5.58 and 5.59 display the percent difference of the lowest score $\%\Delta E_{\text{lowest}}$ compared to $n = 1$ for the BFGS-XEL and the BFGS-XEL (*Hessian only*), respectively. Negative values indicate that the XEL yields better results than the unmodified case. Lower values indicate better performance. For the BFGS-XEL Figure 5.58 illustrates that in all test proteins $n \neq 1$ yields worse results than $n = 1$ on average but one case yields significantly better results than others. In protein hetr when

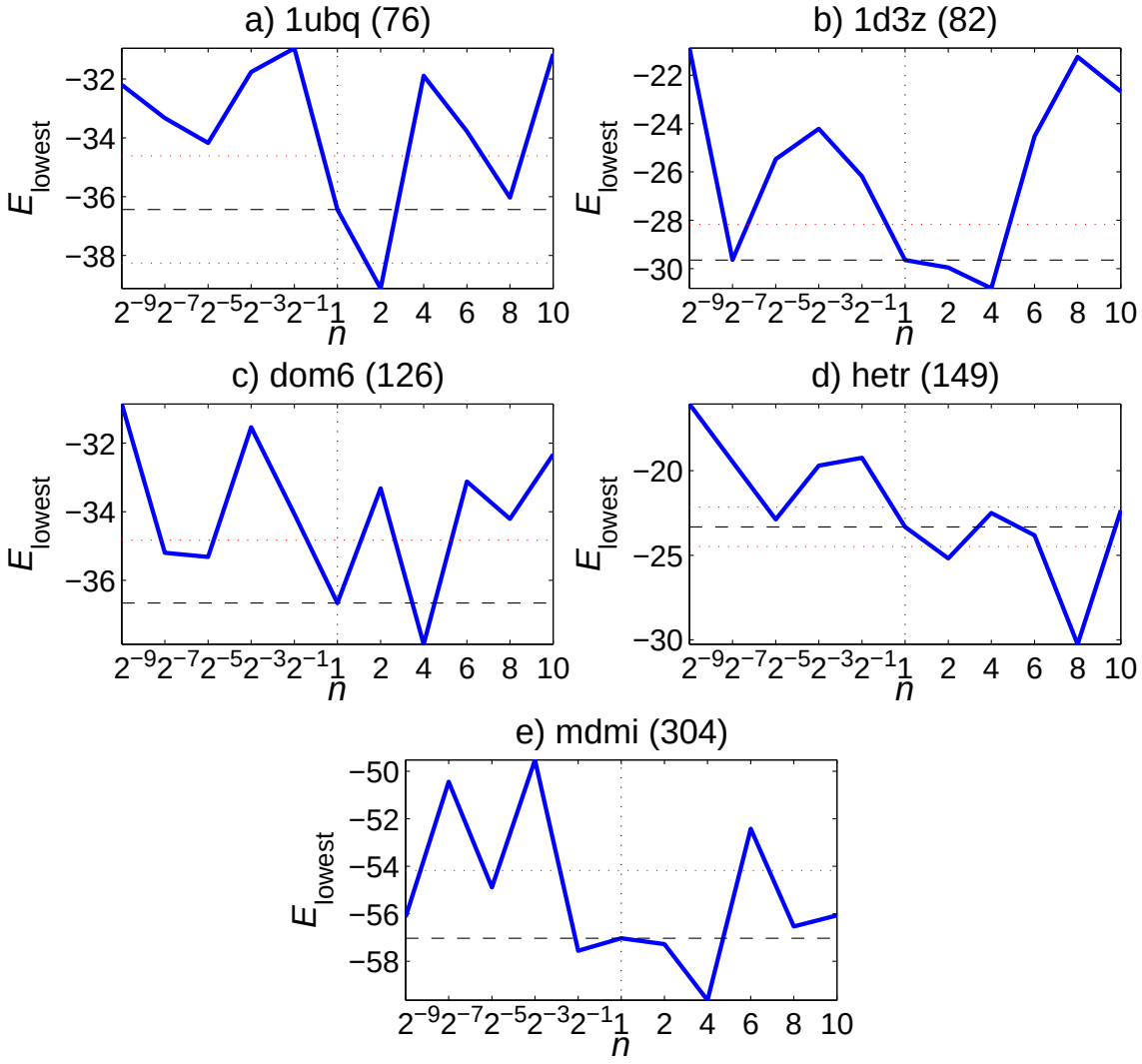

Figure 5.56: BFGS-XEL: The lowest scores (E_{lowest}) versus various values of n in a) 1ubq, b) 1d3z, c) dom6, d) hetr, and e) mdmi. Lower values indicate better performance. The red dotted lines indicate $\pm 5\%$ deviations of the lowest scores from the $n = 1$ cases.

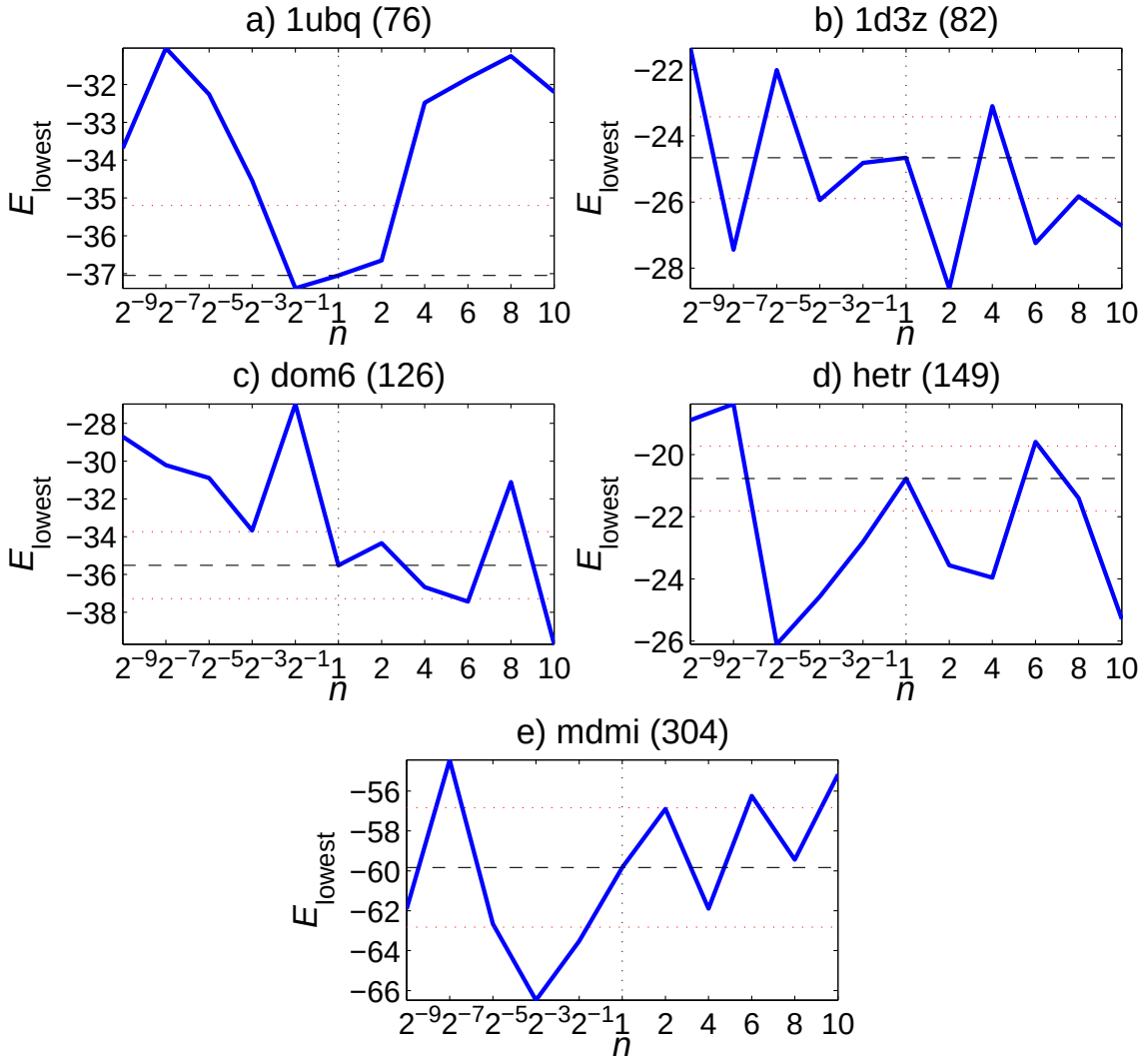

Figure 5.57: BFGS-XEL (*Hessian only*): The lowest scores (E_{lowest}) versus various values of n in a) 1ubq, b) 1d3z, c) dom6, d) hetr, and e) mdmi. Lower values indicate better performance. The red dotted lines indicate $\pm 5\%$ deviations of the lowest scores from the $n = 1$ cases.

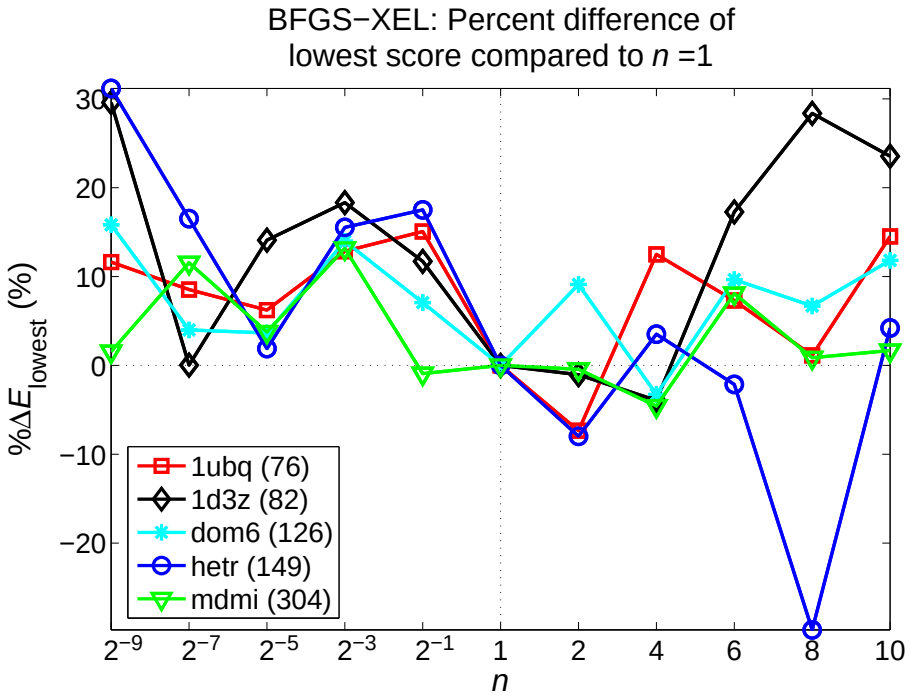

Figure 5.58: BFGS-XEL: The percent difference of the lowest scores ($\%\Delta E_{\text{lowest}}$) compared to $n = 1$ in a) 1ubq, b) 1d3z, c) dom6, d) hetr, and e) mdmi. Lower values indicate better performance.

$n = 8$ the lowest score can be almost 30% lower than that of $n = 1$. The range of $\%\Delta E_{\text{lowest}}$ is between -30% to 31%, most values fall within -10% to 20% range. No correlation between the protein size and the value of $\%\Delta AveSI_n$ can be established. For the BFGS-XEL (*Hessian only*) Figure 5.59 illustrates that in all test proteins $n \neq 1$ yields about the same results than $n = 1$ on average. The worst cases of the BFGS-XEL (*Hessian only*) have better results than those in the BFGS-XEL. The range of $\%\Delta E_{\text{lowest}}$ is between -26% to 25%, most values fall within ±15% range. No correlation between the protein size and the value of $\%\Delta E_{\text{lowest}}$ can be established.

5.2.3.3 Conclusion of lowest score

The lowest score from configurations generated by the QNA-XEL, the QNA-XEL (*Hessian only*), the BFGS-XEL, and the BFGS-XEL (*Hessian only*) with different values of n in proteins 1ubq, 1d3z, dom6, hetr, and mdmi are compared. On average

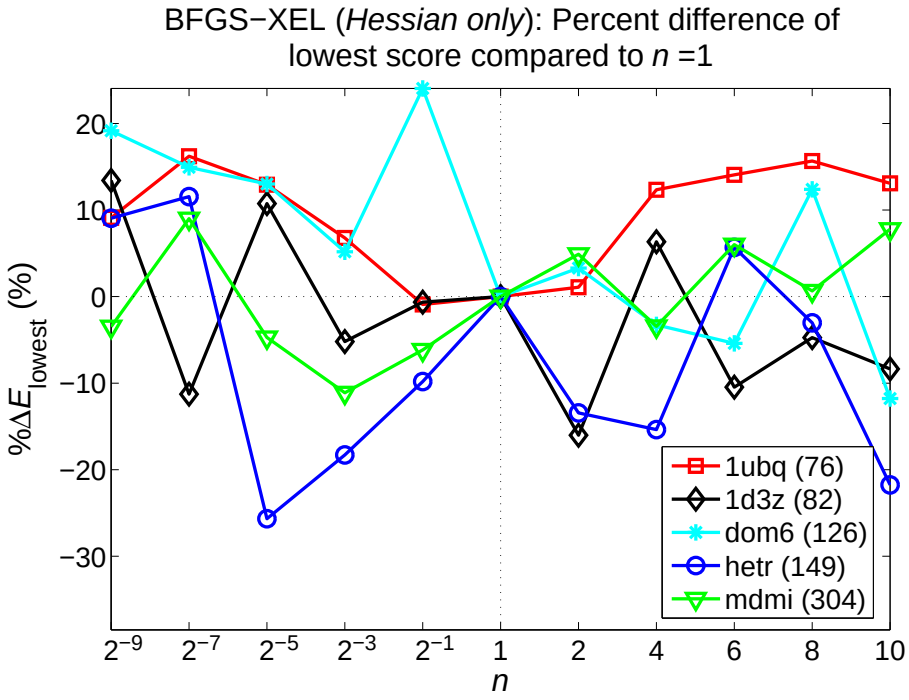

Figure 5.59: BFGS-XEL (*Hessian only*): The percent difference of the lowest scores ($\%\Delta E_{\text{lowest}}$) compared to $n = 1$ in a) 1ubq, b) 1d3z, c) dom6, d) hetr, and e) mdmi. Lower values indicate better performance.

$n \neq 1$ yields higher lowest scores than $n = 1$ which could be up to 38% higher and $n < 1$ yields higher lowest score than $n > 1$.

Table 5.4 summarizes the lowest score for each protein along with the corresponding n value for all test algorithms. All but two cases are the XEL cases, twelve out of all twenty cases are $n > 1$, and five cases are $n = 10$. This indicates a positive effect of the XEL especially with $n > 1$ or $n = 10$ on finding global solutions. Comparing all algorithms, the BFGS-XEL and the BFGS-XEL (*Hessian only*) are the best in finding global solutions as each gives configurations with the lowest score in two proteins.

5.2.4 Average Iterations

For the evaluation of speed, the number of total iterations during the minimization process is determined instead of the actual time of the iterations. This is because Rosetta outputs the time for the entire process while only the time for the analytical

Table 5.4: The n case in which the lowest score of each protein is found.

	Quantities	**1ubq** (76)	**1d3z** (82)	**dom6** (126)	**hetr** (149)	**mdvi** (304)
QNA-XEL	Lowest Score	-38.17	-28.58	-38.32	-25.10	-62.68
	n	10	2^{-3}	10	10	2^{-7}
QNA-XEL (*Hessian only*)	Lowest Score	-39.76	-30.62	-36.34	-29.15	-63.12
	n	1	6	2^{-7}	10	1
BFGS-XEL	Lowest Score	-39.12	-30.82	-37.86	-30.26	-59.64
	n	2	4	4	8	4
BFGS-XEL (*Hessian only*)	Lowest Score	-37.39	-28.61	-39.69	-26.10	-66.48
	n	2^{-1}	2	10	2^{-5}	2^{-3}

minimization is of the interest and it was not possible to separate this out. In addition, the numbers of total iterations are independent of the CPU speed which varies by computer.

Average iteration $AveIt$ is the average of these numbers over the number of decoys N,

$$AveIt = \frac{1}{N}\sum_{i=1}^{N} m_i \tag{5.5}$$

where m_i is the total iterations for the i^{th} decoy. The results from XEL cases are compared to those from unmodified cases by calculating differences between the $AveIt$ values of unmodified cases and those of the XEL cases relative to the $AveIt$ values of unmodified cases. The percent difference of the average iteration compared to unmodified case $\%\Delta AveIt$ is defined as

$$\%\Delta AveIt_n = \frac{AveIt_n - AveIt_{n=1}}{|AveIt_{n=1}|} \times 100 \tag{5.6}$$

where $AveIt_n$ is the $AveIt$ value of the XEL case and $AveIt_{n=1}$ is the $AveIt$ value of unmodified case. A negative $\%\Delta AveIt_n$ indicates that the XEL case has a smaller $AveIt$ value than the unmodified case and therefore a better result. This means simulations in the XEL case run on average with fewer total iterations.

5.2.4.1 QNA-XEL and QNA-XEL (*Hessian only*)

The $AveIt$ from different values of n in proteins 1ubq, 1d3z, dom6, hetr, and mdmi are compared in Figures 5.60 and 5.61 for the QNA-XEL and the QNA-XEL (*Hessian only*), respectively. A lower value implies higher speed. The red dotted lines above and below black dashed horizontal lines indicate 5% deviations above and below the average iteration of the $n = 1$ case. Some red dotted lines may not be visible if all data lie inside the upper or lower boundary. Some black dashed lines may not be visible if the data points for $n = 1$ are at the higher ends of the data ranges.

For the QNA-XEL Figure 5.60 shows that in most cases the XEL requires fewer iterations than the unmodified case. On average $n < 1$ cases require slightly fewer iterations than $n > 1$ cases. When $n < 1$ $AveIt$ decreases as n decreases but when $n < 2^{-3}$ it decreases at a lower rate. When $n > 1$ $AveIt$ decreases as n increases. Almost all proteins have the maximum $AveIt$ when $n = 1$ but protein hetr has the maximum $AveIt$ when $n = 2$. For the QNA-XEL (*Hessian only*) Figure 5.61 shows results similar to the QNA-XEL but when $n > 1$ $AveIt$ decreases at a slower rate as n increases. For almost all proteins, the maximum $AveIt$ is at $n = 2$, but for protein 1ubq the maximum $AveIt$ is at $n = 1$.

The $\%\Delta AveIt_n$ from different values of n in proteins 1ubq, 1d3z, dom6, hetr, and mdmi are compared in Figures 5.62 and 5.63 for the QNA-XEL and the QNA-XEL (*Hessian only*), respectively. A lower value implies higher speed. For the QNA-XEL Figure 5.62 demonstrates no correlation between the size of proteins and the $\%\Delta AveIt_n$ values. The results also show that the XEL can reduce the number of iterations by up to 40% when $n < 1$ and up to 30% when $n > 1$. For the QNA-XEL (*Hessian only*) Figure 5.63 shows results similar to the QNA-XEL but when $n > 1$ the $\%\Delta AveIt_n$ is higher than that in the QNA-XEL on average. The results also show that the XEL can reduce the number of iterations by up to 47% when $n < 1$ and up to 22% when $n > 1$.

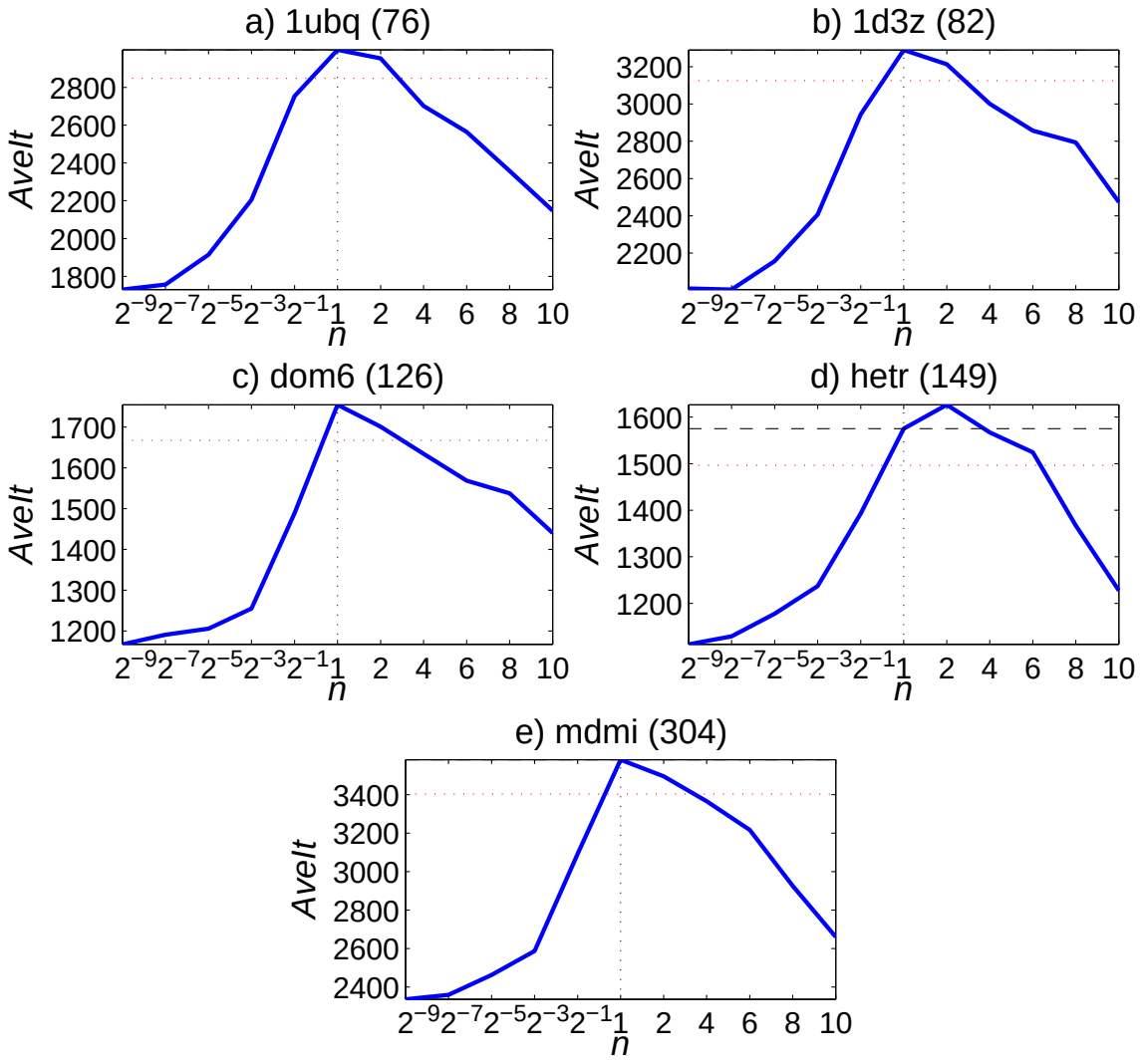

Figure 5.60: QNA-XEL: Average iterations ($AveIt$) versus various values of n in a) 1d3z, b) 1ubq, c) dom6, d) hetr, and e) mdmi. Lower numbers mean simulations run with fewer iterations. The red dotted lines above and below black dashed horizontal lines indicate 5% deviations above and below the average iteration of the $n = 1$ case.

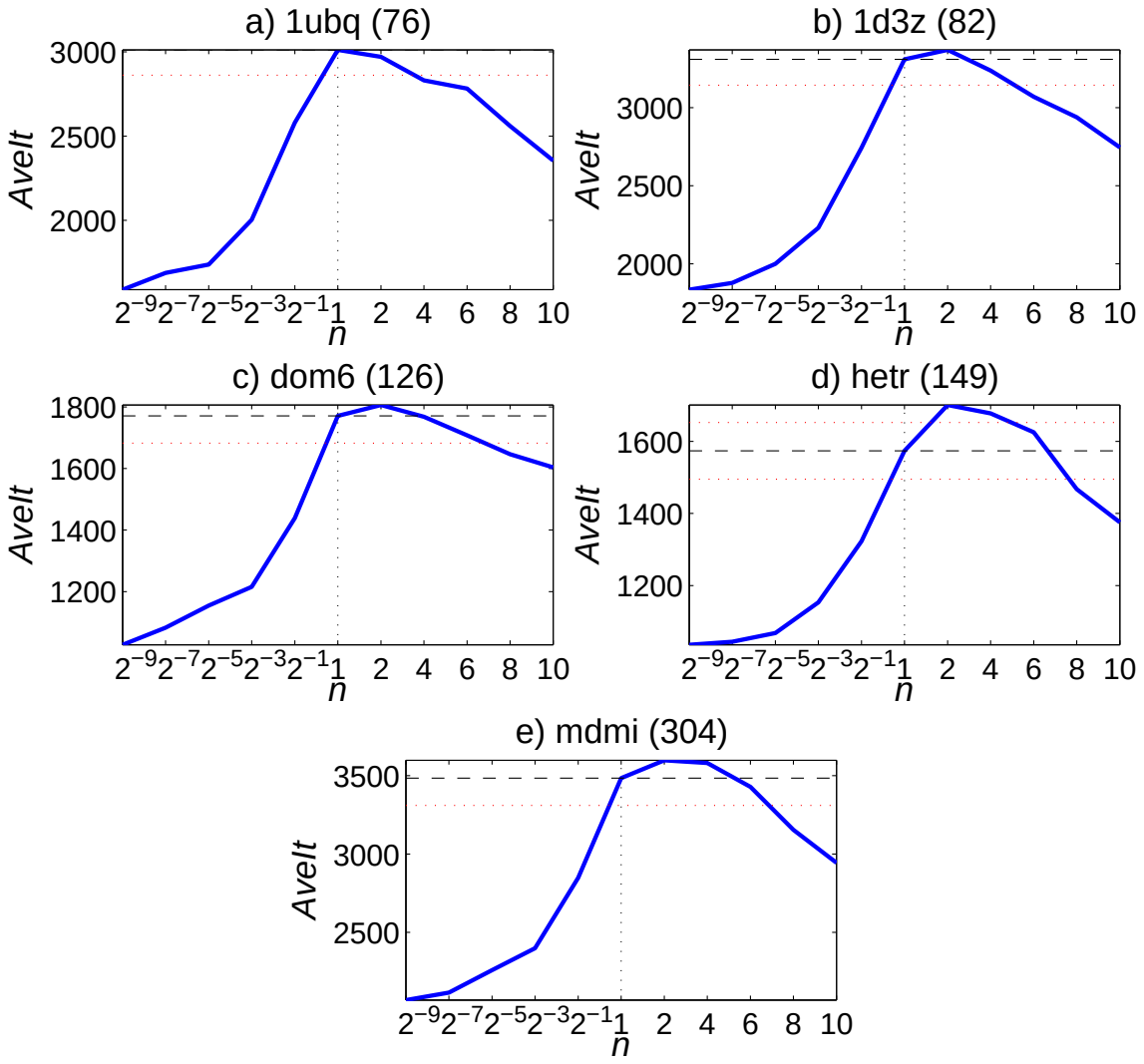

Figure 5.61: QNA-XEL (*Hessian only*): Average iterations ($AveIt$) versus various values of n in a) 1d3z, b) 1ubq, c) dom6, d) hetr, and e) mdmi. Lower numbers mean simulations run with fewer iterations. The red dotted lines above and below black dashed horizontal lines indicate 5% deviations above and below the average iteration of the $n = 1$ case.

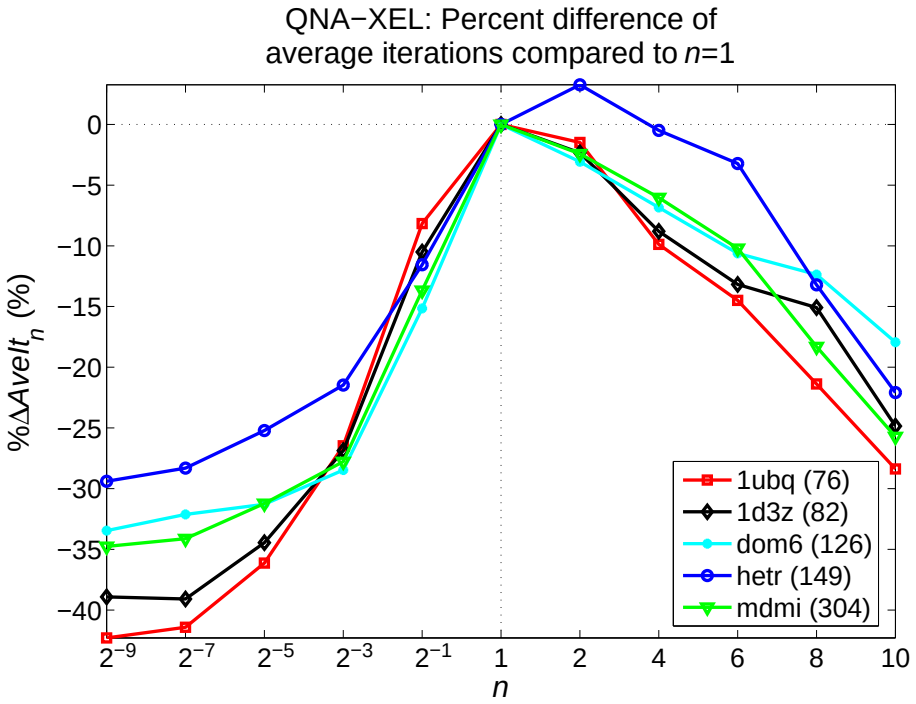

Figure 5.62: QNA-XEL: The percent difference of average iterations ($\%\Delta AveIt_n$) compared to the unmodified case ($n = 1$) versus various values of n. Lower values mean simulations run with fewer average iterations.

5.2.4.2 BFGS-XEL and BFGS-XEL (Hessian only)

The $AveIt$ from different values of n in proteins 1ubq, 1d3z, dom6, hetr, and mdmi are compared in Figures 5.64 and 5.65 for the BFGS-XEL and the BFGS-XEL (*Hessian only*), respectively. A lower value implies higher speed. The red dotted lines above and below black dashed horizontal lines indicate 5% deviations above and below the average iteration of the $n = 1$ case. Some red dotted lines may not be visible if all data lie inside the upper or lower boundary. Some black dashed lines may not be visible if the data points for $n = 1$ are at the higher ends of the data ranges.

For the BFGS-XEL Figure 5.64 shows that in most cases the XEL requires fewer iterations than the unmodified case. On average $n < 1$ cases require slightly fewer iterations than $n > 1$ cases. When $n < 1$ $AveIt$ decreases as n decreases, but when $n > 1$ $AveIt$ decreases as n increases. Almost all proteins have the maximum $AveIt$

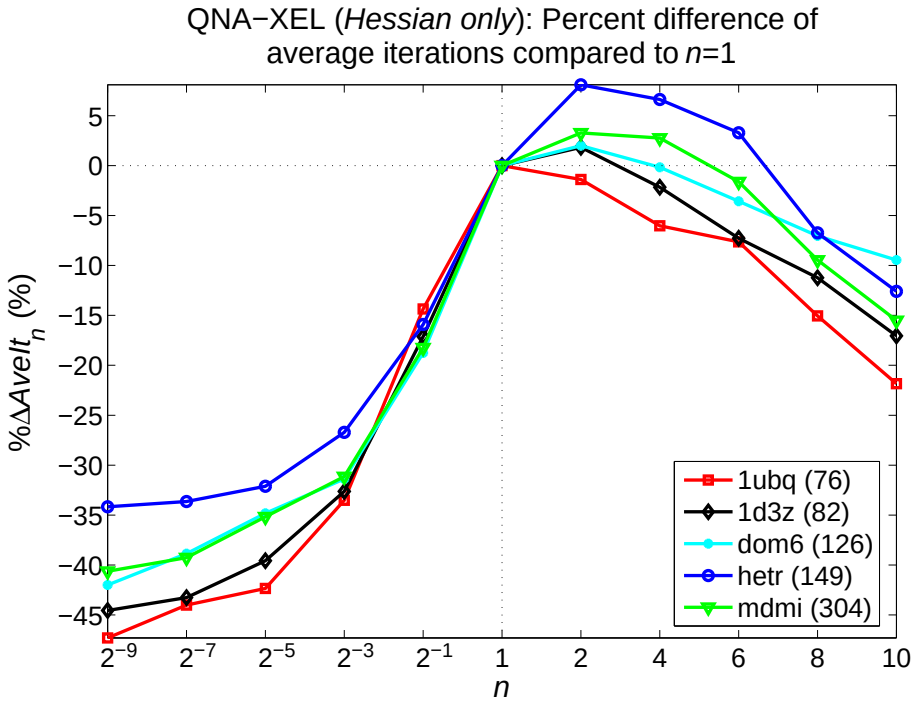

Figure 5.63: QNA-XEL (*Hessian only*): The percent difference of average iterations ($\%\Delta AveIt_n$) compared to the unmodified case ($n = 1$) versus various values of n. Lower values mean simulations run with fewer average iterations.

when $n = 2$ but protein 1ubq has a maximum $AveIt$ when $n = 1$. For the BFGS-XEL (*Hessian only*) Figure 5.65 shows results similar to the BFGS-XEL but when $n > 1$ $AveIt$ decays at a slower rate as n increases. For smaller proteins with the number of residues less than 100, 1ubq and 1d3z, the maximum $AveIt$'s are at $n = 1$, but for the larger proteins the maximum are at $n = 2$. For protein mdmi when $n > 1$ $AveIt$ is higher than $n = 1$.

The $\%\Delta AveIt_n$ from different values of n in proteins 1ubq, 1d3z, dom6, hetr, and mdmi are compared in Figures 5.66 and 5.67 for the BFGS-XEL and the BFGS-XEL (*Hessian only*), respectively. A lower value implies higher speed. For the BFGS-XEL Figure 5.66 demonstrates some correlation between the size of proteins and the $\%\Delta AveIt_n$ values. In most n cases, the graphs of 1d3z and 1ubq, which have the smallest numbers of residues, frequently exhibit the most improvement, and the graph of mdmi, which has the largest number of residues, frequently exhibits the

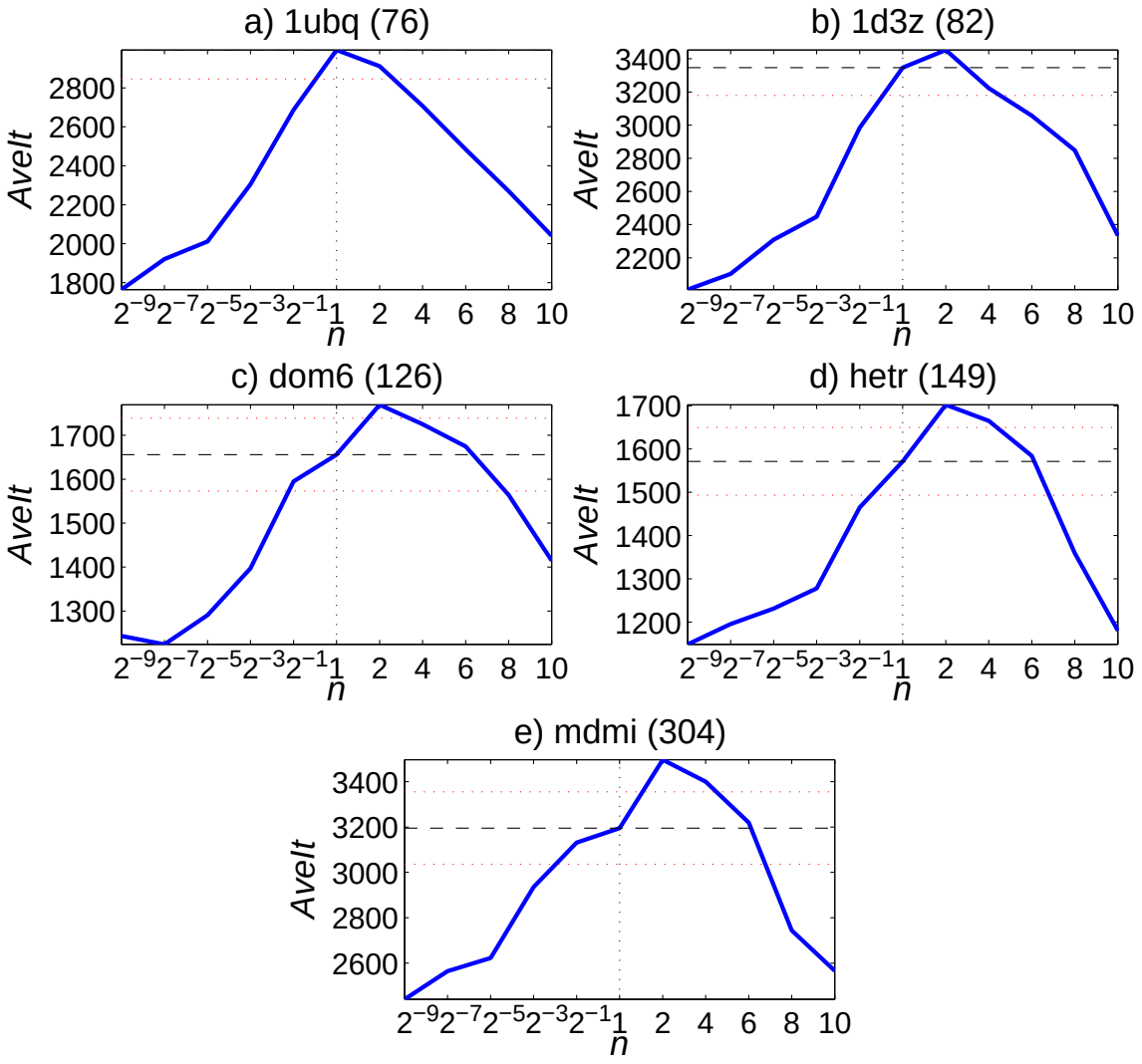

Figure 5.64: BFGS-XEL: Average iterations ($AveIt$) versus various values of n in a) 1d3z, b) 1ubq, c) dom6, d) hetr, and e) mdmi. Lower numbers mean simulations run with fewer iterations. The red dotted lines above and below black dashed horizontal lines indicate 5% deviations above and below the average iteration of the $n = 1$ case.

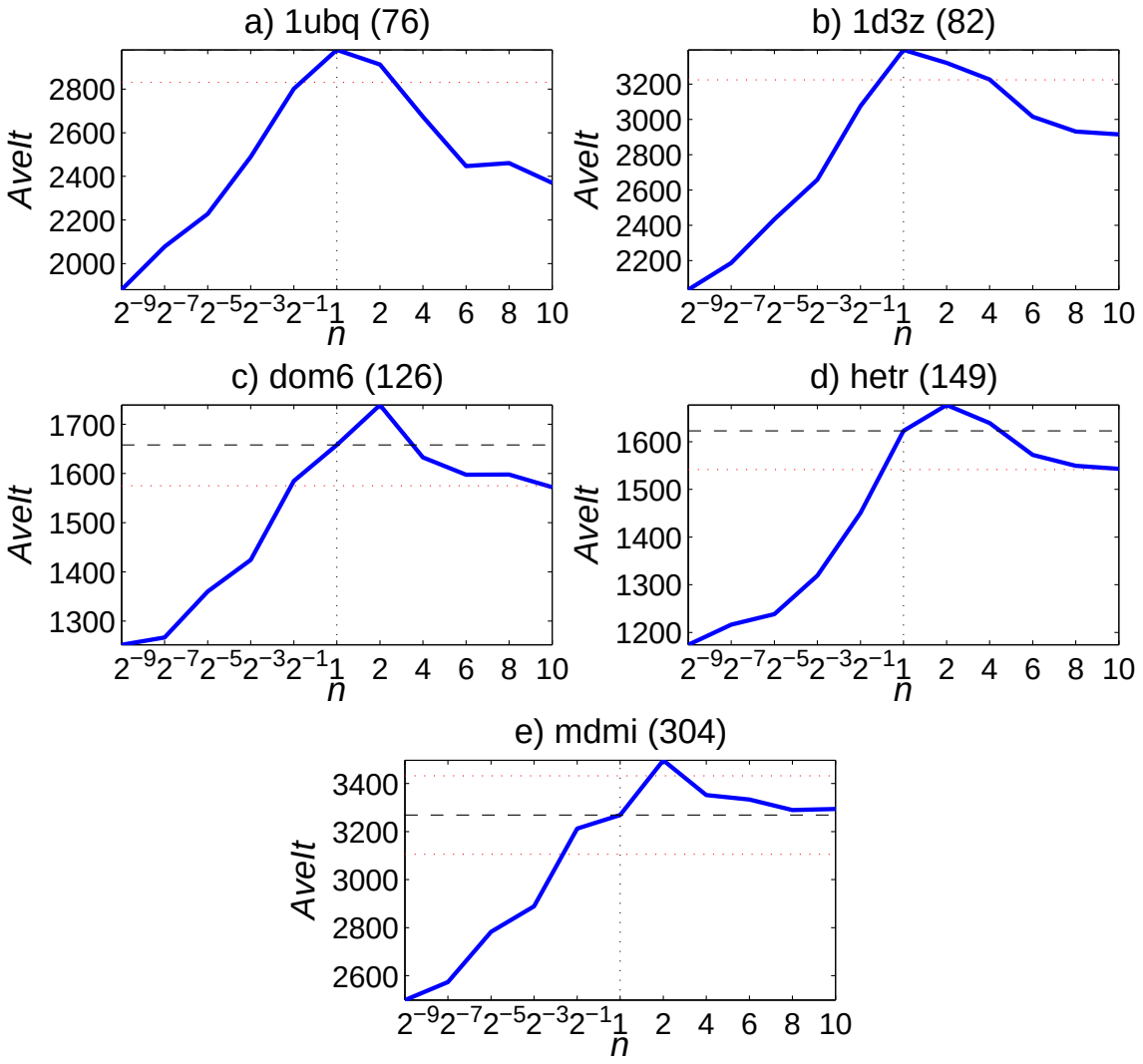

Figure 5.65: BFGS-XEL (*Hessian only*): Average iterations ($AveIt$) versus various values of n in a) 1d3z, b) 1ubq, c) dom6, d) hetr, and e) mdmi. Lower numbers mean simulations run with fewer iterations. The red dotted lines above and below black dashed horizontal lines indicate 5% deviations above and below the average iteration of the $n = 1$ case.

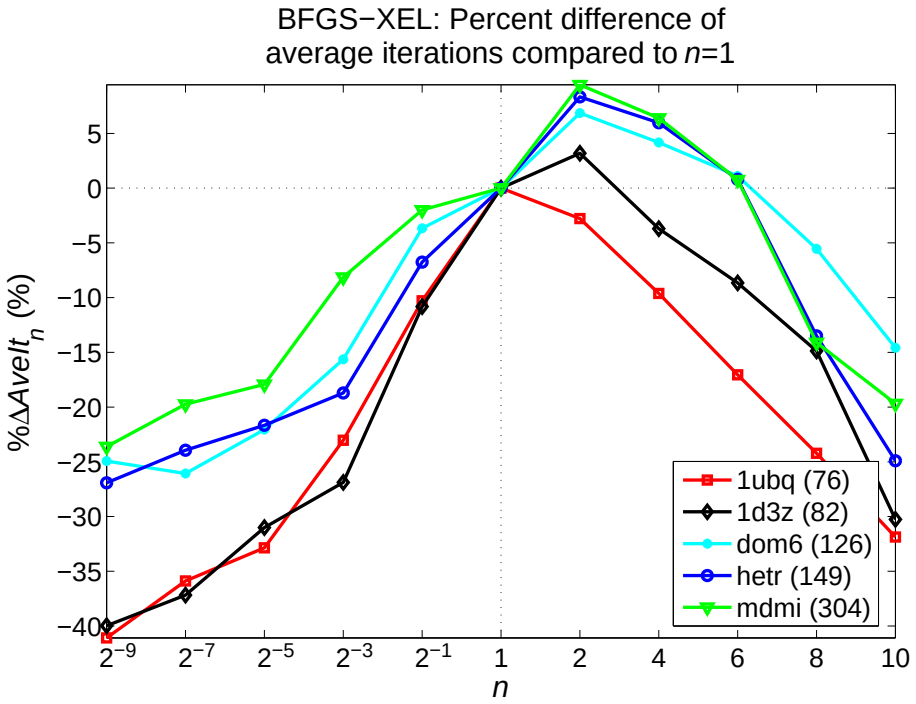

Figure 5.66: BFGS-XEL: The percent difference of average iterations ($\%\Delta AveIt_n$) compared to the unmodified case ($n = 1$) versus various values of n. Lower values mean simulations run with fewer average iterations.

least improvement. This suggests $\%\Delta AveIt_n$ values may increase as the protein sizes increase. The results also show that the XEL can reduce the number of iterations by up to 40% when $n < 1$ and up to 34% when $n > 1$. For the BFGS-XEL (*Hessian only*) Figure 5.67 shows results similar to the BFGS-XEL but when $n = 8$ and 10 the $\%\Delta AveIt_n$ are higher than those in the BFGS-XEL. The results also show that the XEL can reduce the number of iterations by up to 40% when $n < 1$ and up to 20% when $n > 1$.

5.2.4.3 Conclusion of average iterations

Results from all algorithms are quite similar as most cases have a lower number of iterations than $n = 1$ and $n < 1$ has a lower number of iterations than $n > 1$. This implies that the XEL results in higher speed with $n < 1$ yielding higher speed than $n > 1$ and that the XEL has consistent effects on the number of iterations

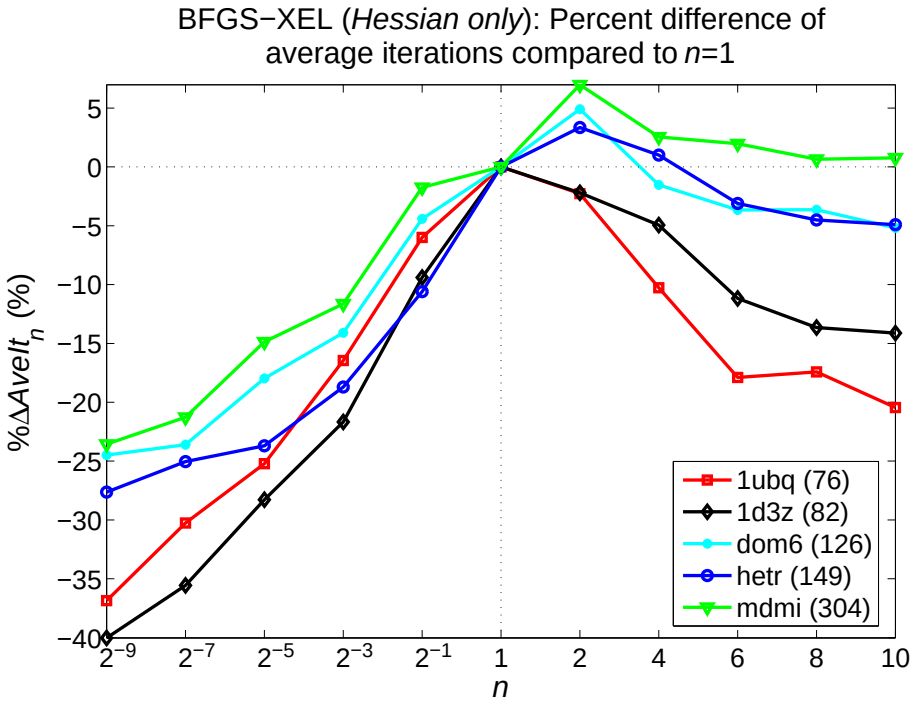

Figure 5.67: BFGS-XEL (*Hessian only*): The percent difference of average iterations ($\%\Delta AveIt_n$) compared to the unmodified case ($n = 1$) versus various values of n. Lower values mean simulations run with fewer average iterations.

in the QNA-XEL, the QNA-XEL (*Hessian only*), the BFGS-XEL, and the BFGS-XEL (*Hessian only*). However, this finding is not consistent with that in Section 5.1.1 which shows that with the bracketing and Brent line search the QNA-XEL, the QNA-XEL (*Hessian only*), the BFGS-XEL, and the BFGS-XEL (*Hessian only*) have slightly higher speed when $n > 1$. This inconsistency may be because the simulated energy landscapes do not represent the score function of a protein molecule.

Reducing the number of iterations by up to 40% roughly means reducing the runtime by 40% because the runtime for the BFGS and the BFGS-XEL with $n = 1$ are roughly the same. This is very beneficial since the protein conformation prediction process can take several months even for a small protein. For example, a high-resolution structure prediction algorithm [11] refining 20,000 to 30,000 models of several 49 to 88 residue long molecules takes about 150 CPU days per molecule. 40% reduction means the runtime reduces by 60 CPU days.

5.2.5 Efficiency

To consider both score improvement and total iterations in the performance evaluation, efficiency, e_n, is defined as the value of average score improvement per thousand iterations,

$$e_n = \frac{AveSI_n}{AveIt_n/1000} \tag{5.7}$$

Since negative $AveSI_n$'s mean results are improved from input decoys, the negative efficiency indicates the improvement rate and lower values mean higher rates. In all simulations presented here the $AveSI$s are negative so all values for e are negative.

The percent differences between the values of efficiency in the various values of n and those in the unmodified cases, $\%\Delta e_n$, are displayed in Figure 5.75,

$$\%\Delta e_n = \frac{e_n - e_{n=1}}{|e_{n=1}|} \times 100 \tag{5.8}$$

where $e_{n=1}$ is the efficiency of the unmodified case. Lower values indicate that the XEL cases have better performance than the unmodified case. This means the same number of iterations in XEL-implemented simulations yields more score improvement.

5.2.5.1 QNA-XEL and QNA-XEL (Hessian only)

The efficiency e_n from different values of n in proteins 1ubq, 1d3z, dom6, hetr, and mdmi are compared in Figures 5.68 and 5.69 for the QNA-XEL and the QNA-XEL (*Hessian only*), respectively. A lower value implies better efficiency. The red dotted lines indicate $\pm 5\%$ deviations of the efficiency from the $n = 1$ cases indicated by black dashed horizontal lines. Some red dotted lines may not be visible if all data lie inside the upper or lower boundary. Some black dashed lines may not be visible if the data points for $n = 1$ are at the higher ends of the data ranges.

For the QNA-XEL Figure 5.68 shows that efficiency is dependent on the size of the protein as it decreases when the protein chain length increases. This means a larger protein results in better efficiency. The better efficiency in a larger protein is because

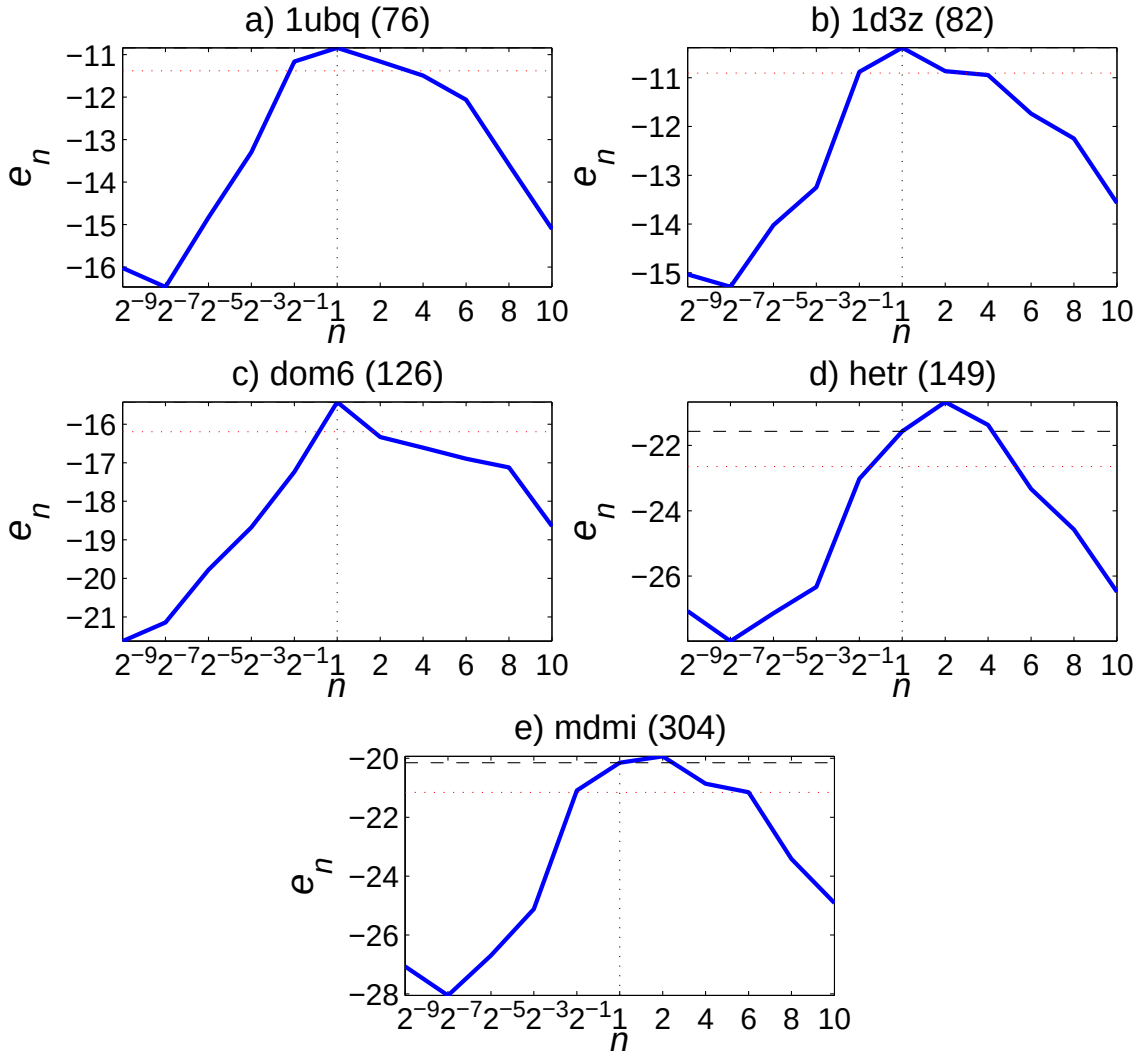

Figure 5.68: QNA-XEL: The efficiency (e_n) versus various values of n in a) 1ubq, b) 1d3z, c) dom6, d) hetr, and e) mdmi. Lower values mean better performance. The red dotted lines above and below black dashed horizontal lines indicate 5% deviations above and below the efficiency of the $n = 1$ case.

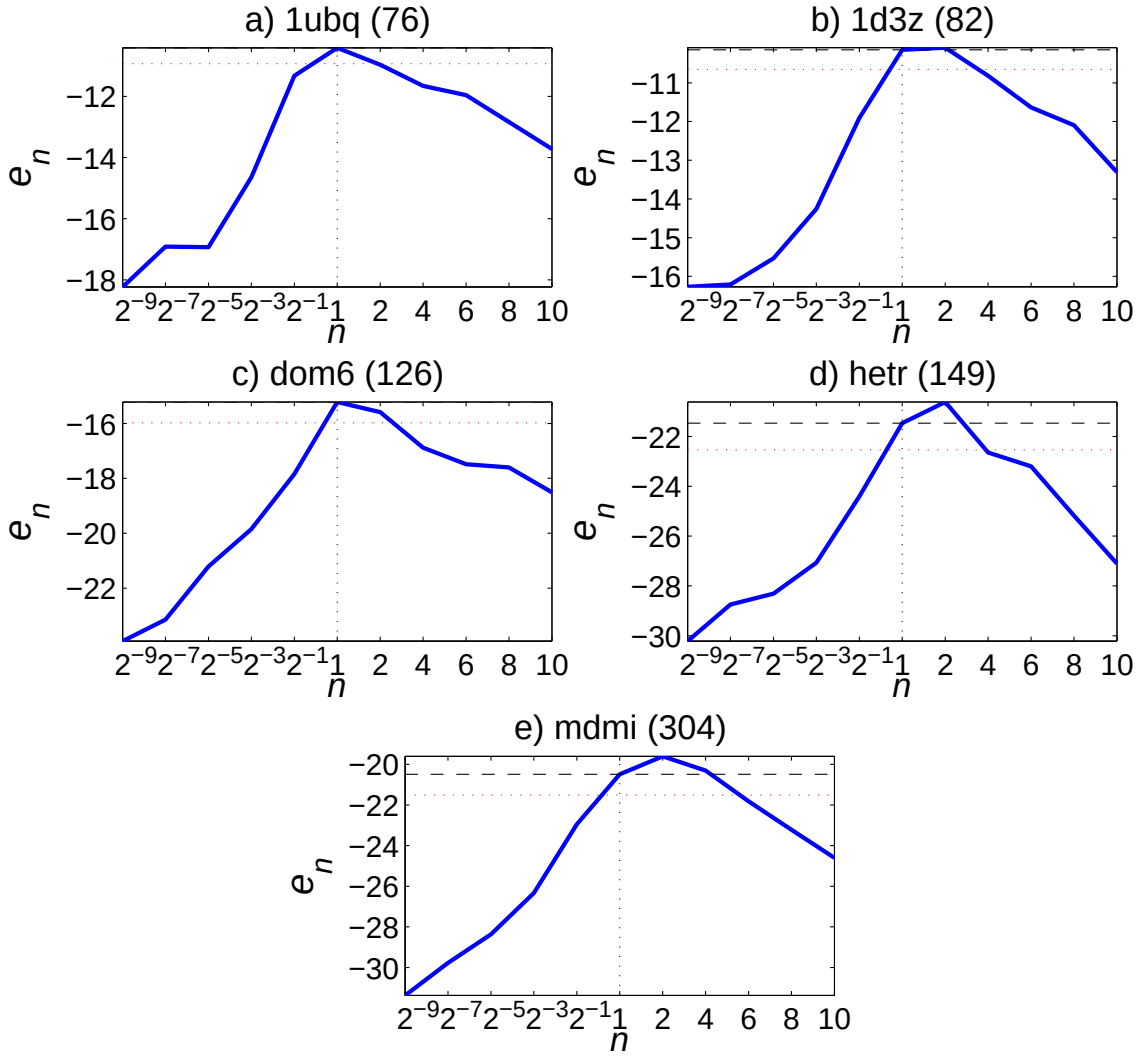

Figure 5.69: QNA-XEL (*Hessian only*): The efficiency (e_n) versus various values of n in a) 1ubq, b) 1d3z, c) dom6, d) hetr, and e) mdmi. Lower values mean better performance. The red dotted lines above and below black dashed horizontal lines indicate 5% deviations above and below the efficiency of the $n = 1$ case.

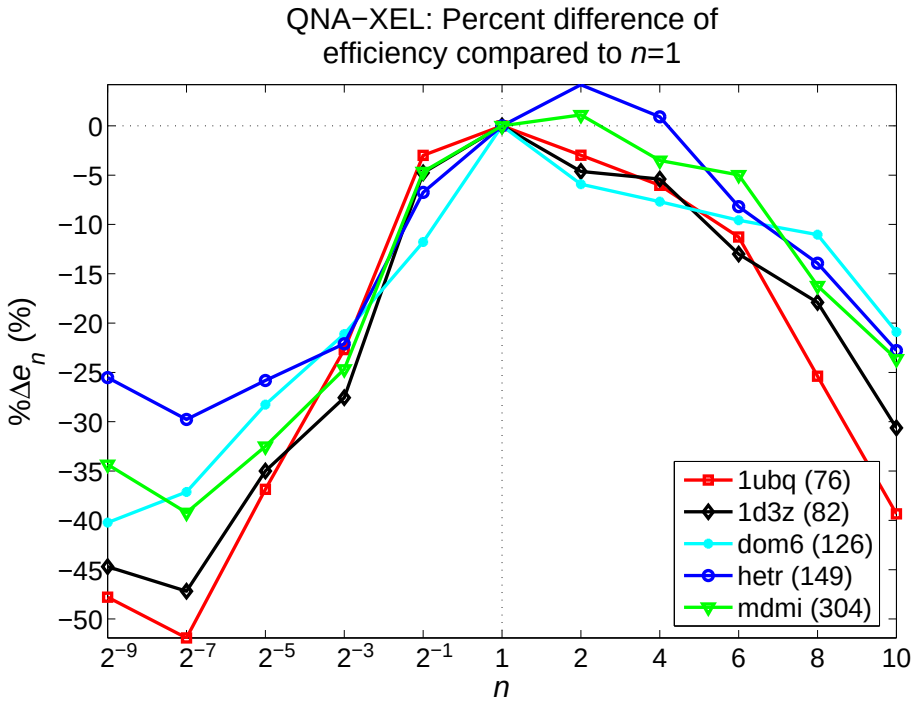

Figure 5.70: QNA-XEL: The percent difference of the efficiency ($\%\Delta e_n$) compared to those in the unmodified cases ($n = 1$) versus various values of n. Lower values mean better performance.

a larger gap of scores between an input decoy and a semi-folded configuration. An input decoy of a larger protein is more likely to be a bad approximation because of a larger conformational space and thus a refinement protocol can significantly improve the score which yields better efficiency. On average $n < 1$ cases have better efficiency than $n > 1$. When $n < 1$ efficiency is improved as n decreases but not at $n = 2^{-9}$. When $n > 1$ efficiency is also improved as n increases but at a lower rate. Proteins 1ubq, 1d3z, and dom6 have the worst efficiency when $n = 1$ and proteins hetr and mdmi have the worst efficiency when $n = 2$. For the QNA-XEL (*Hessian only*) Figure 5.69 show results similar to those in the QNA-XEL but with better efficiency.

The $\%\Delta e_n$ from different values of n in proteins 1ubq, 1d3z, dom6, hetr, and mdmi are compared in Figures 5.70 and 5.71 for the QNA-XEL and the QNA-XEL (*Hessian only*), respectively. A lower value implies better efficiency.

For the QNA-XEL Figures 5.70 shows that the efficiency can be up to 52% better

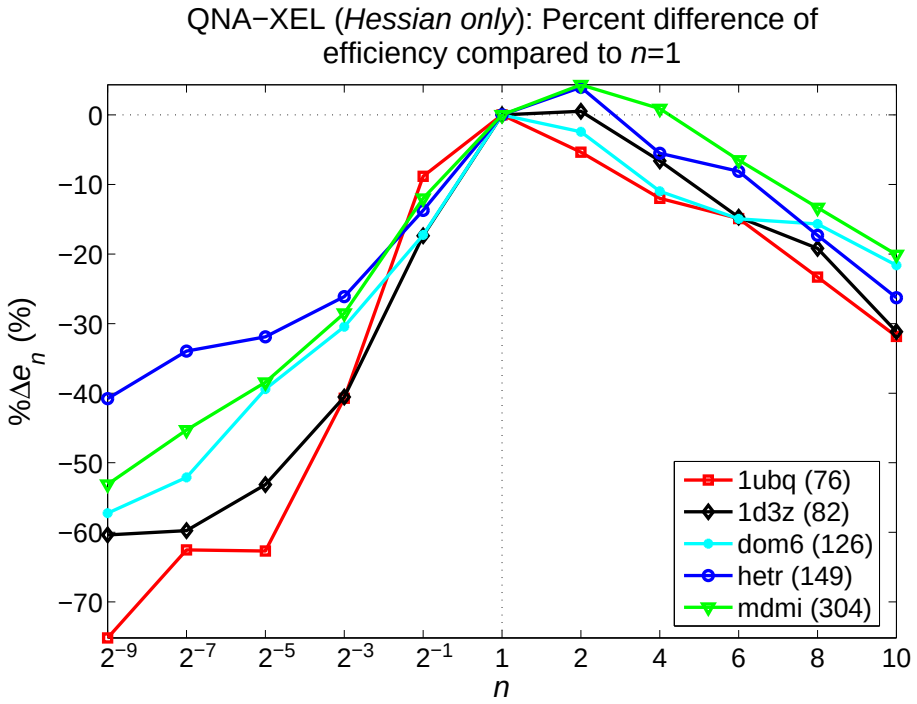

Figure 5.71: QNA-XEL (*Hessian only*): The percent difference of the efficiency ($\%\Delta e_n$) compared to those in the unmodified cases ($n = 1$) versus various values of n. Lower values mean better performance.

than that of $n = 1$ which means with the same number of iterations as $n = 1$ the score improvement can be lower by 50%. When $n > 1$ the efficiency can be up to 40% better than that of $n = 1$. No correlation between the size of proteins and the values of $\%\Delta e_n$ can be found but a smaller protein tends to have higher improvement in efficiency than a larger protein. For the QNA-XEL (*Hessian only*) Figures 5.71 shows results similar to those of the QNA-XEL but efficiency can be up to 75% better than that of $n = 1$. When $n > 1$ efficiency can be up to 30% better than that of $n = 1$. Some correlation between the size of proteins and the values of $\%\Delta e_n$ can be seen as a smaller protein has higher improvement in efficiency than a larger protein when $n < 2^{-1}$ or $n > 6$ except for protein hetr.

5.2.5.2 BFGS-XEL and BFGS-XEL (*Hessian only*)

The e_n from different values of n in proteins 1ubq, 1d3z, dom6, hetr, and mdmi are compared in Figures 5.72 and 5.73 for the BFGS-XEL and the BFGS-XEL (*Hessian only*), respectively. A lower value implies better efficiency. The red dotted lines indicate $\pm 5\%$ deviations of efficiency from the $n = 1$ cases indicated by black dashed horizontal lines. Some red dotted lines may not be visible if all data lie inside the upper or lower boundary. Some black dashed lines may not be visible if the data points for $n = 1$ are at the higher ends of the data ranges.

For the BFGS-XEL Figure 5.72 shows that the value of efficiency is dependent on the size of the protein as the value decreases when the protein chain length increases. On average the $n < 1$ case has better efficiency than the $n > 1$ case in proteins 1ubq, 1d3z, and dom6 but both cases have about the same efficiency in proteins hetr and mdmi. When $n < 1$ efficiency is improved as n decreases but when $n > 1$ efficiency is improved as n increases. The worst efficiency is when $n = 1$ or 2. For the BFGS-XEL (*Hessian only*) Figure 5.73 show results slightly different from those of the BFGS-XEL. While in the BFGS-XEL (*Hessian only*) efficiency is also improved as the protein chain length increases, $n < 1$ case has much better efficiency than $n > 1$ case.

The $\%\Delta e_n$ from different values of n in proteins 1ubq, 1d3z, dom6, hetr, and mdmi are compared in Figures 5.74 and 5.75 for the BFGS-XEL and the BFGS-XEL (*Hessian only*), respectively. A lower value implies better efficiency.

For the BFGS-XEL Figure 5.74 shows that efficiency can be up to 50% better than that of $n = 1$ and when $n > 1$ efficiency can be up to 35% better than that of $n = 1$. No correlation between the size of proteins and the values of $\%\Delta e_n$ can be found but a smaller protein tends to have better efficiency than a larger protein. For the BFGS-XEL (*Hessian only*) Figure 5.75 shows results similar to those of the BFGS-XEL but efficiency can be up to 52% better than that of $n = 1$ and when

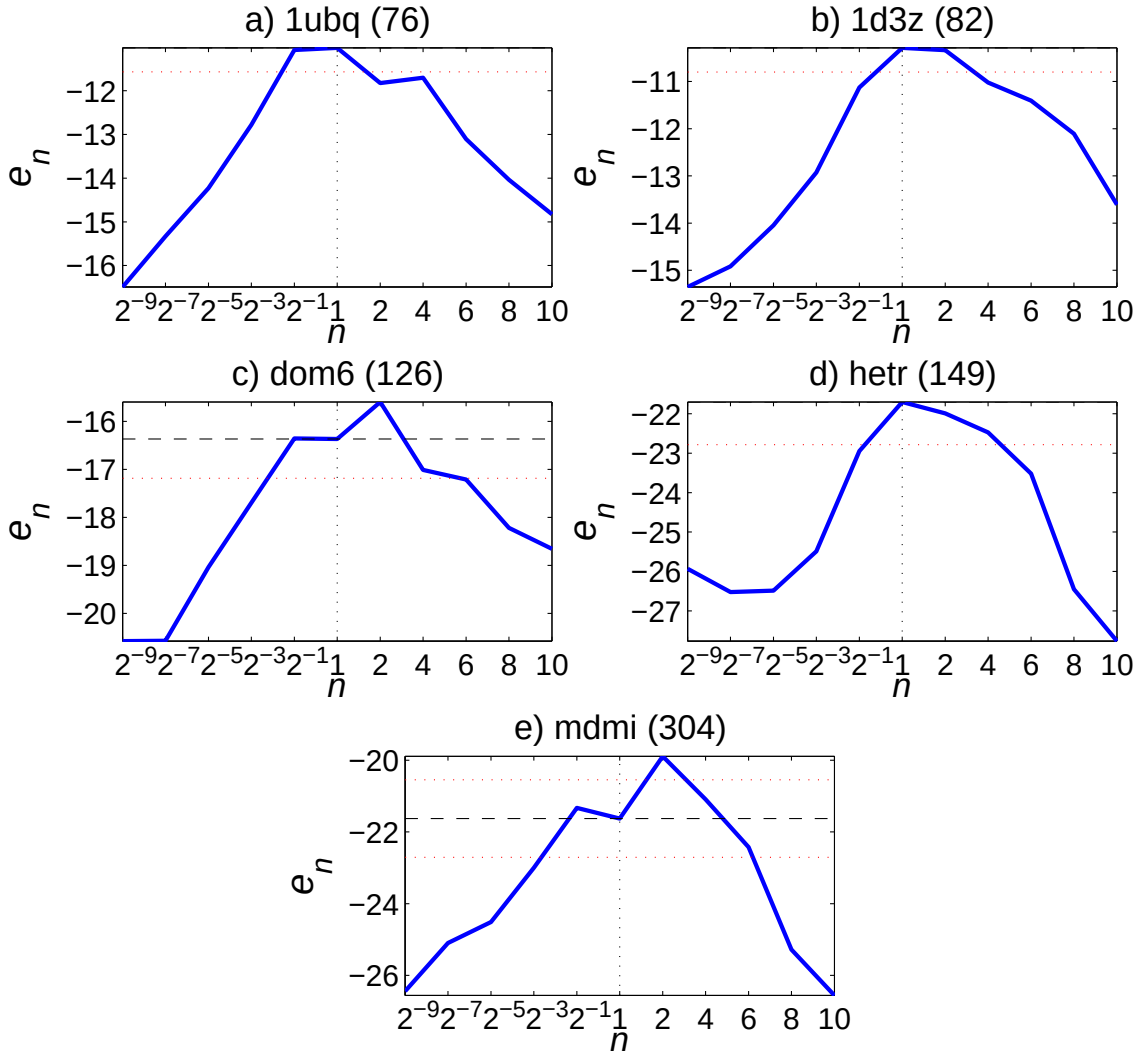

Figure 5.72: BFGS-XEL: efficiency (e_n) versus various values of n in a) 1ubq, b) 1d3z, c) dom6, d) hetr, and e) mdmi. Lower values mean better performance. The red dotted lines above and below black dashed horizontal lines indicate 5% deviations above and below the efficiency of the $n = 1$ case.

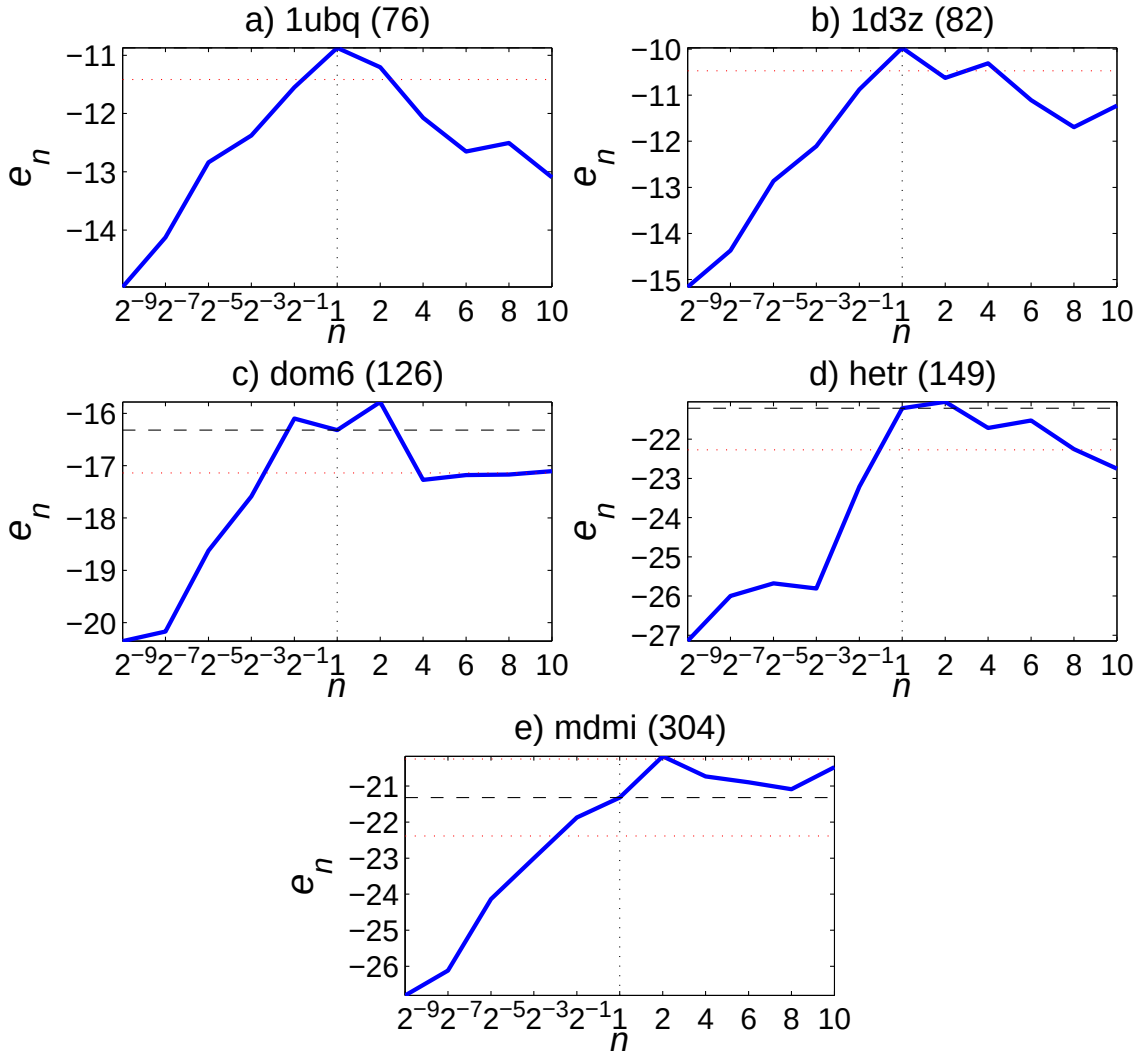

Figure 5.73: BFGS-XEL (*Hessian only*): efficiency (e_n) versus various values of n in a) 1ubq, b) 1d3z, c) dom6, d) hetr, and e) mdmi. Lower values mean better performance. The red dotted lines above and below black dashed horizontal lines indicate 5% deviations above and below the efficiency of the $n = 1$ case.

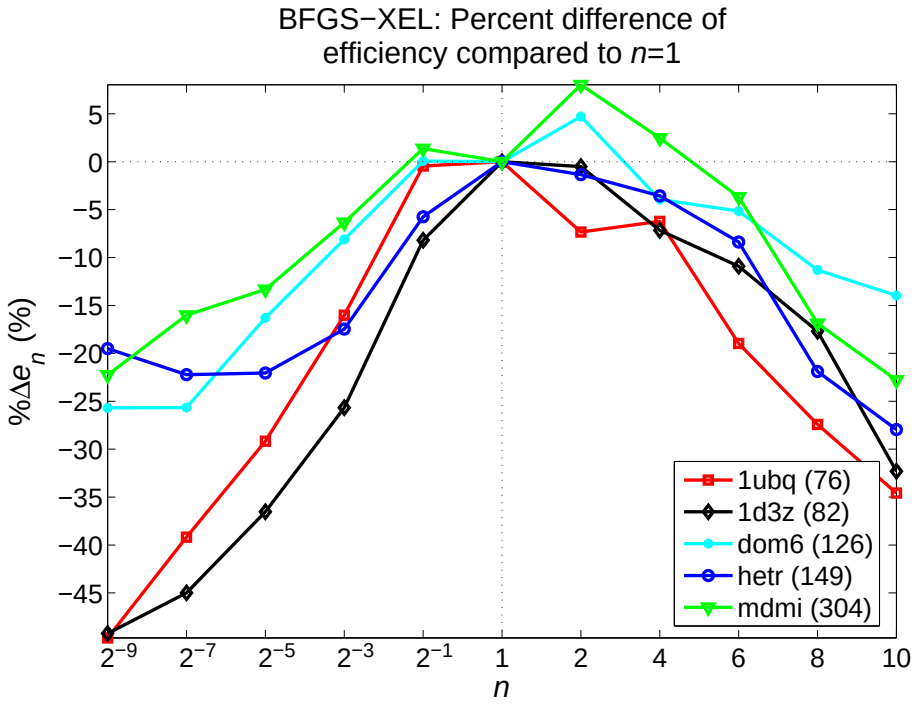

Figure 5.74: BFGS-XEL: The percent difference of the efficiency ($\%\Delta e_n$) compared to those in the unmodified cases ($n = 1$) versus various values of n. Lower values mean better performance.

$n > 1$ efficiency can be up to 20% better than that of $n = 1$. No correlation between the size of proteins and the values of $\%\Delta e_n$ can be seen but a smaller protein tends to have better efficiency than a larger protein.

5.2.5.3 Conclusion of efficiency

Results show that the XEL can improve the efficiency in the QNA-XEL by up to 52%, in the QNA-XEL (*Hessian only*) by up to 75%, in the BFGS-XEL by up to 50%, and in the BFGS-XEL (*Hessian only*) by up to 52%. In all algorithms efficiency is dependent on the size of the protein as it is improved when the protein chain length increases. On average when $n < 1$ the efficiency is better than when $n \geq 1$ which implies that better efficiency can be achieved by using $n < 1$.

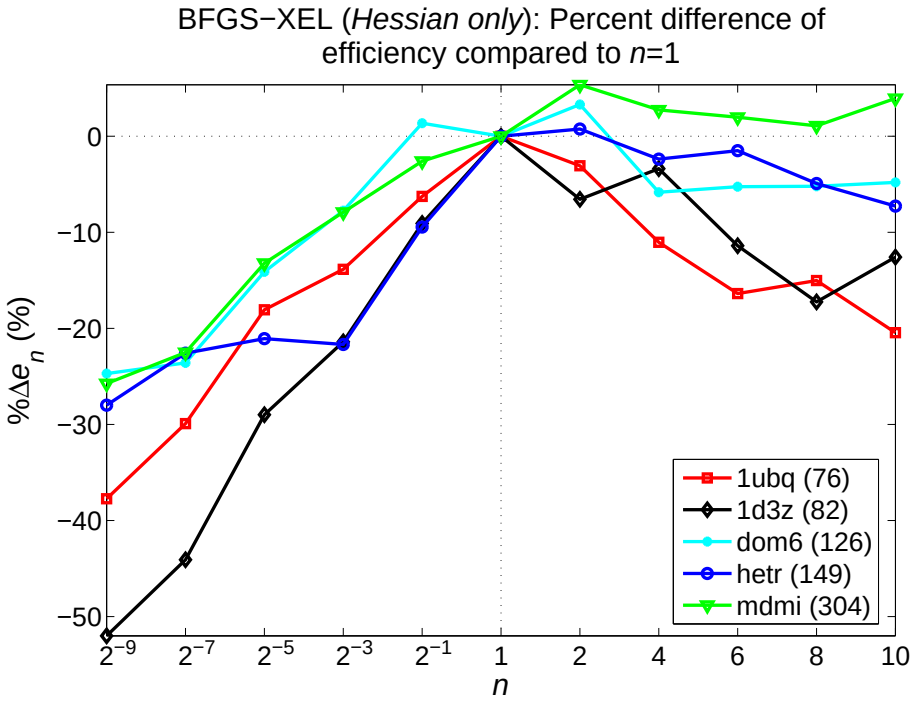

Figure 5.75: BFGS-XEL (*Hessian only*): The percent difference of the efficiency ($\%\Delta e_n$) compared to those in the unmodified cases ($n = 1$) versus various values of n. Lower values mean better performance.

5.2.6 Similarity to the native configuration

In addition to quality, speed, and efficiency, the similarity of output configurations to the native conformation are evaluated in terms of TM score [88], GDT score [79, 80], MaxSub score [62], and RMSD [32, 33] using TM-score software package [86]. The RMSD (Relative Mean Square Deviation) value commonly used to evaluate the similarity is usually not sufficient [88] because the value depends on the protein length and percent alignment which is the compared portions of the output configuration and the native conformation. The TM, the GDT, and the MaxSub scores are developed to improve a correlation between high similarity and a high score. These scores are range between 0 and 1 and a value closer to 1 implies a configuration more similar to the native state. The TM score has a better correlation than the GDT and the MaxSub scores [88].

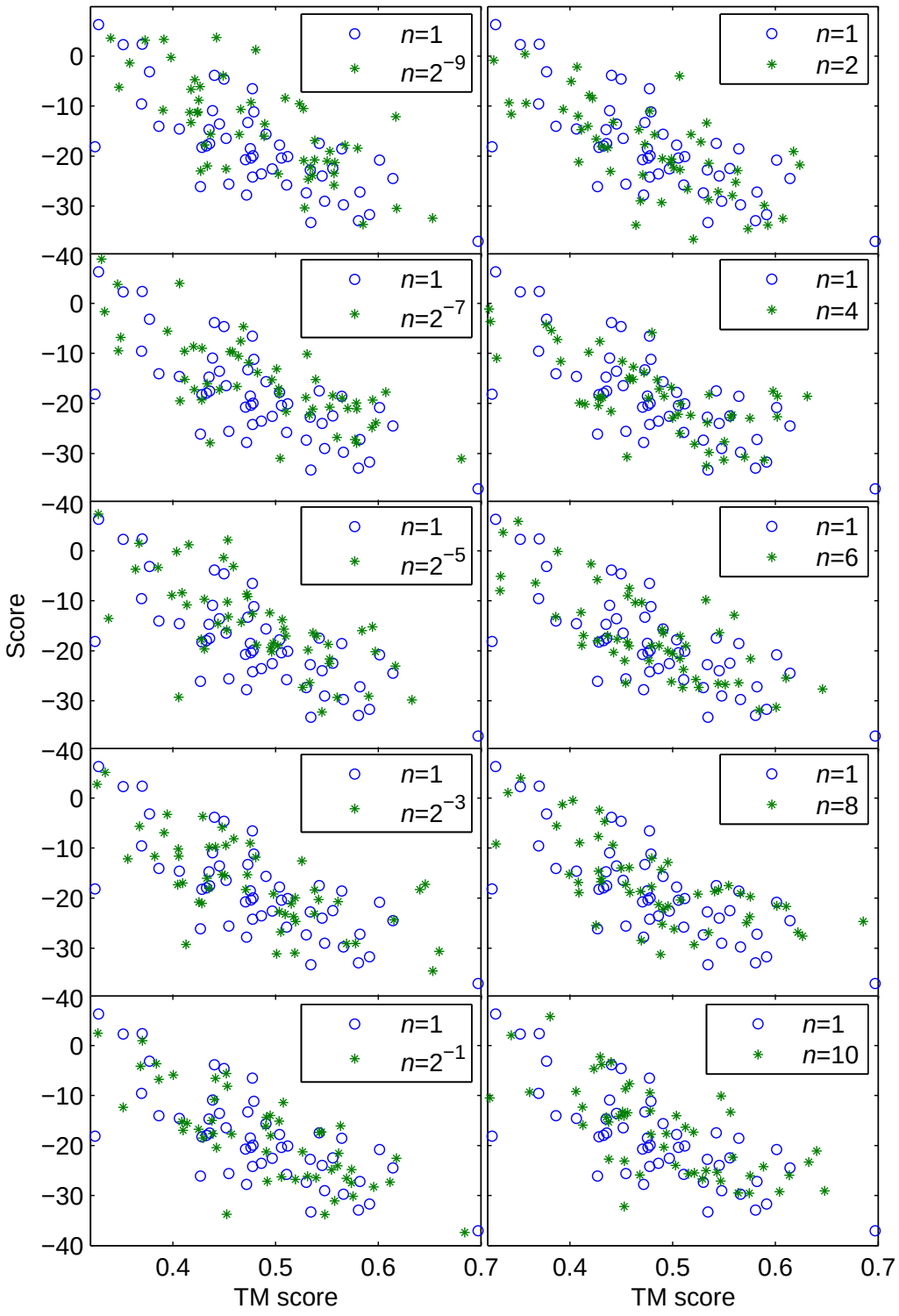

Figure 5.76: BFGS-XEL (*Hessian only*): The scores versus TM scores of resulting configurations with various values of n for protein 1ubq. Lower score means better quality. TM score closer to 1 means a configuration more similar to the native conformation.

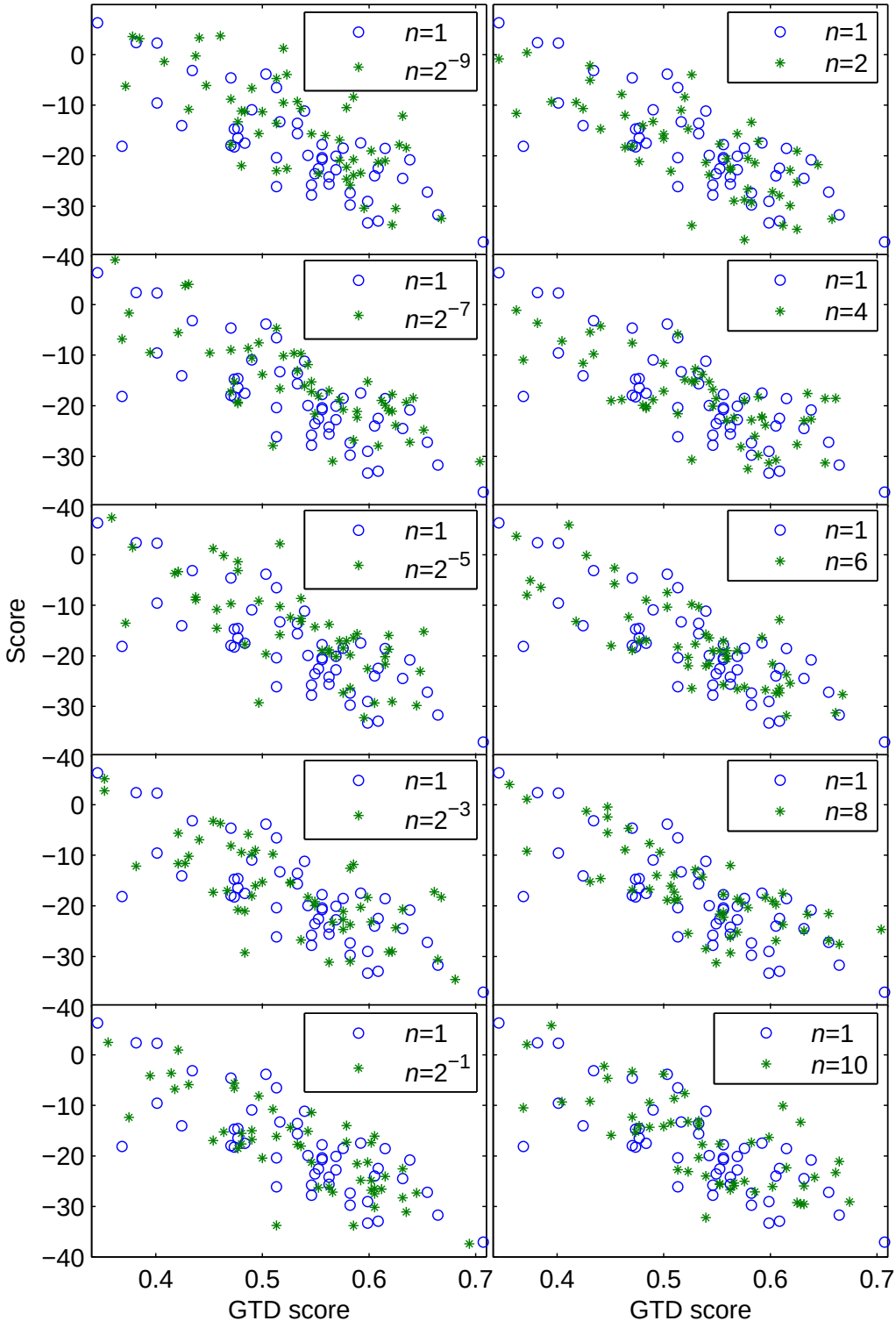

Figure 5.77: BFGS-XEL (*Hessian only*): The scores versus GTD scores of resulting configurations with various values of n for protein 1ubq. Lower score means better quality. GTD score closer to 1 means a configuration more similar to the native conformation.

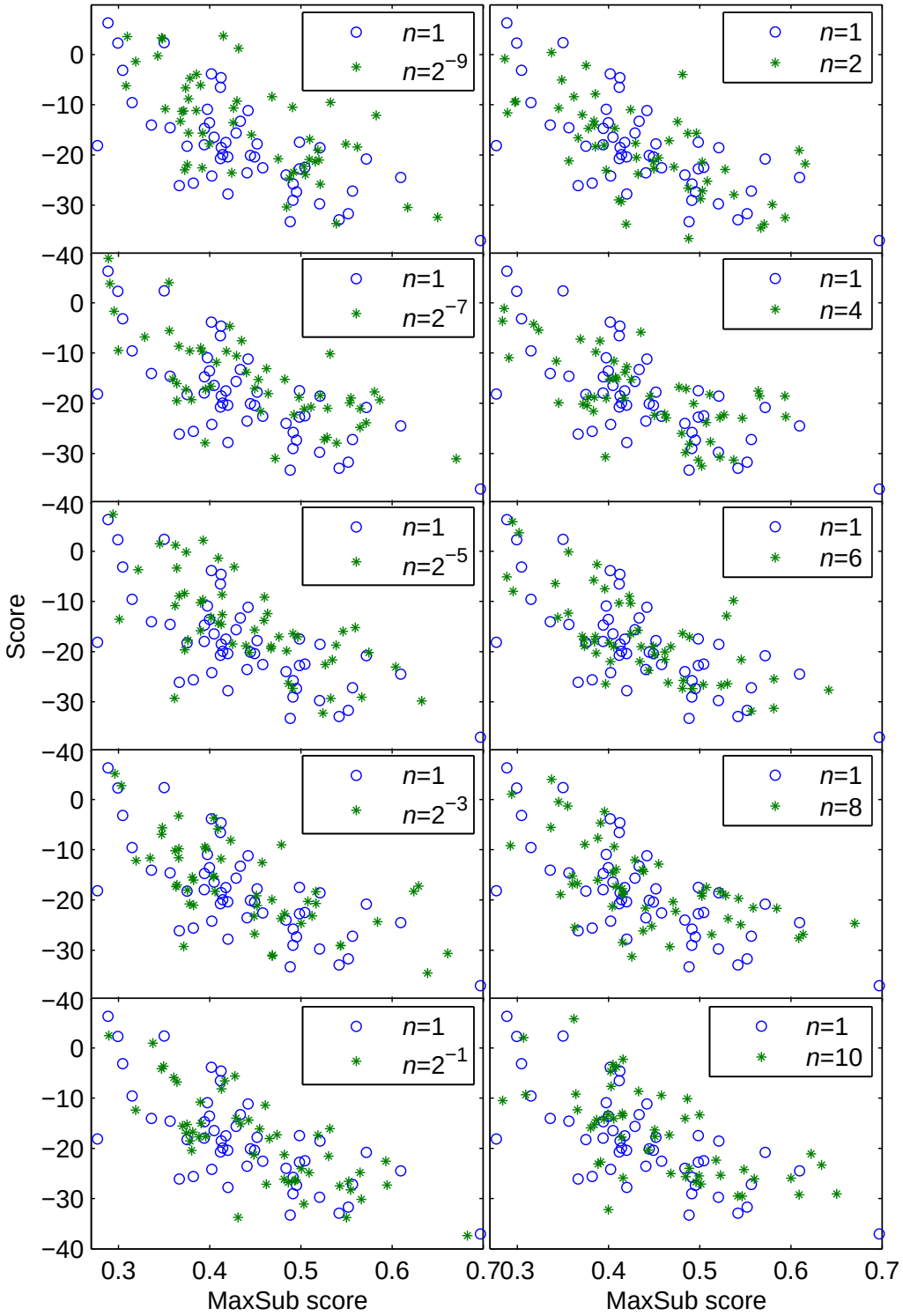

Figure 5.78: BFGS-XEL (*Hessian only*): The scores versus MaxSub scores of resulting configurations with various values of n for protein 1ubq. Lower score means better quality. MaxSub score closer to 1 means a configuration more similar to the native conformation.

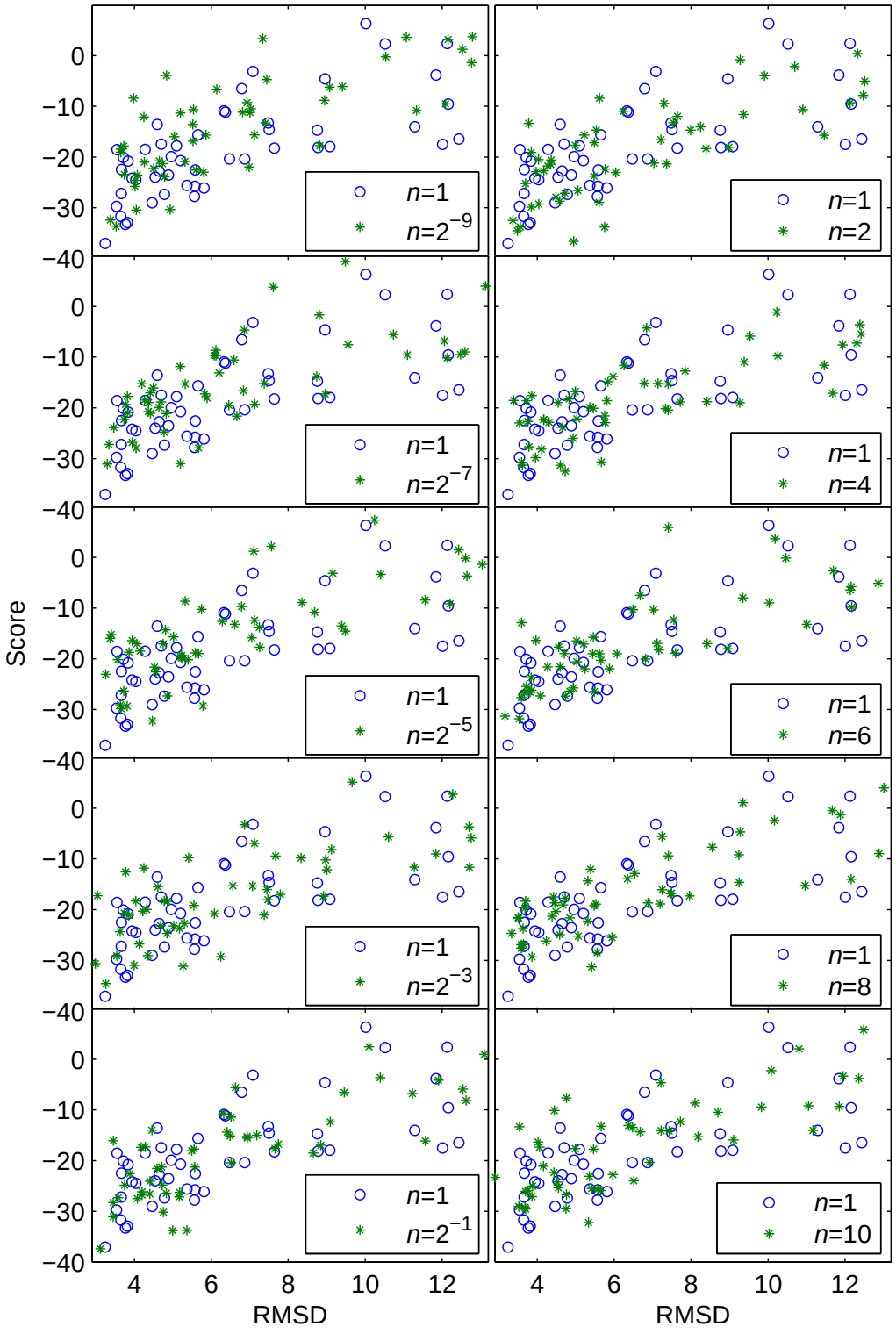

Figure 5.79: BFGS-XEL (*Hessian only*): The scores versus RMSD values of resulting configurations with various values of n for protein 1ubq. Lower score means better quality. RMSD value closer to 0 means a configuration more similar to the native conformation.

Figures 5.76 through 5.79 present the TM scores, the GDT scores, the MaxSub scores, and the RMSD values of the resulting configurations of protein 1ubq with BFGS-XEL (*Hessian only*) and various values of n. Results from each value of n (shown as stars) are compared to those from $n = 1$ (shown as circles). A TM, a GDT, or a MaxSub score closer to 1 means a configuration more similar to the native conformation. A TM score greater than 0.4 implies a meaningful prediction. An RMSD value closer to 0 means a configuration more similar to the native conformation. Note that only protein 1ubq is evaluated because other proteins do not have the structural information of native conformations available.

Results in Figures 5.76 through 5.79 show that the TM scores, the GDT scores, the MaxSub scores, and the RMSD values from $n \neq 1$ and $n = 1$ are similar since the circles ($n = 1$) and the stars ($n \neq 1$) cover similar areas. This means $n \neq 1$ yields configurations similar to the native conformation as those of $n = 1$. Figures 5.76 through 5.78 show that as n increases the stars move slightly downward and to the right. This implies that the quality of resulting configurations is improved as n increases.

5.2.6.1 Conclusion of similarity to the native configuration

Overall results show that XEL has little effect on the similarity to the native configuration. Since XEL yields better speed, this means XEL would be expected to yield configurations with the same quality as those of the unmodified case in terms of similarity with fewer iterations.

5.2.7 Further analysis: Same-iteration comparison

The XEL algorithms yield lower quality than the unmodified cases as their average score improvement $AveSI$ and lowest score E_{lowest} are typically higher than those for $n = 1$. However, these quantities could possibly be improved if XEL simulations are allowed to run with the same number of total iterations from all decoys as the

unmodified cases but with more perturbations or decoys. For example, in protein 1ubq running simulations with 29 additional decoys yields the same number of total iterations as the $n = 1$ case. This claim is investigated on the BFGS-XEL (*Hessian only*) with $n = 2^{-9}$ and two proteins, 1ubq and dom6. As additional simulations are run, more decoys (Decoys+) or more perturbations (Perturb+) are added so that the resulting number of total iterations are nearly equal to the $n = 1$ case (less than 1% difference). The results from the runs with more decoys and perturbations are compared to the results from the BFGS-XEL (*Hessian only*) with $n = 2^{-9}$ presented in previous sections and referred to below as the $n = 2^{-9}$ run without additional iterations or $n = 2^{-9}$ (50) in figures.

Figures 5.80 presents a) the percent difference of the average score improvement $\%\Delta AveSI$, b) the percent difference of the lowest score $\%\Delta E_{lowest}$, c) the percent difference of the average iteration $\%\Delta AveIt$, and d) the percent difference of the efficiency $\%\Delta e_n$ from special runs with more decoys (Decoys+) and more perturbations (Perturb+) and the BFGS-XEL (*Hessian only*) without additional iterations. All runs use $n = 2^{-9}$ and are performed for proteins 1ubq and dom6. Lower percentage implies better performance.

Figures 5.80 a) shows that more decoys does not improve the values of $\%\Delta AveSI_n$ in the case of either protein but more perturbations improves the values of $\%\Delta AveSI_n$ in the case of both proteins by 9% and 6% from those in the $n = 2^{-9}$ run without additional iterations. This shows that the values of $\%\Delta AveSI_n$ is sensitive to the average number of total iterations per decoy ($AveIt$). Since adding more decoys does not change the $AveIt$ value (Figure 5.80 c) and the values of $\%\Delta AveSI_n$ in the $n = 2^{-9}$ run without additional iterations is possibly a good representative value for that $AveIt$, more decoys has almost no effect on the values of $\%\Delta AveSI_n$. With more perturbations the values of $\%\Delta AveSI_n$ are improved since adding more perturbations increases the $AveIt$.

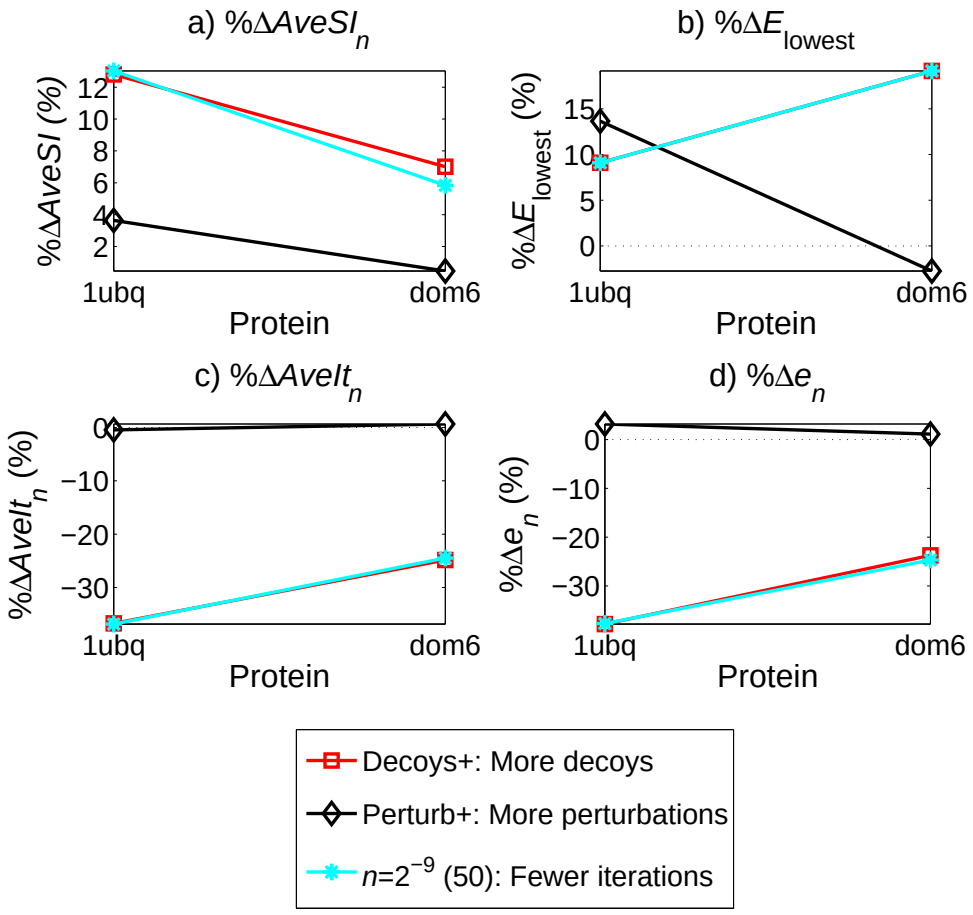

Figure 5.80: Same-Iteration: a) the percent difference of average score improvement $\%\Delta AveSI$, b) the percent difference of lowest score $\%\Delta E_{lowest}$, c) the percent difference of average iteration $\%\Delta AveIt$, and d) the percent difference of the efficiency $\%\Delta e_n$ from special runs with $n = 2^{-9}$ and either more decoys (Decoys+) or more perturbations (Perturb+) and the BFGS-XEL (*Hessian only*) without additional iterations. All runs use $n = 2^{-9}$ and are performed for proteins 1ubq and dom6. Lower percentage means better results.

Figure 5.80 b) shows that more decoys does not improve or worsen the values of $\%\Delta E_{\text{lowest}}$ in the case of either protein which suggests that adding more decoys would not change the lowest score. This result agrees with an additional test that runs 1000 decoys with $n = 2^{-9}$ and 614 decoys of $n = 1$. Results also show that more perturbations improves the value of $\%\Delta E_{\text{lowest}}$ in protein dom6 by 21% but worsens the value of $\%\Delta E_{\text{lowest}}$ in protein 1ubq by 5% from those in the $n = 2^{-9}$ run without additional iterations. Increasing the number of perturbations can yield a lower or a higher lowest score than those of the $n = 2^{-9}$ run without additional iterations because of the probabilistic nature of the perturbations. Unlike the more perturbation case with new simulations on all original 50 decoys, the more decoy case adds new simulations on additional 29 decoys to the $n = 2^{-9}$ run with original 50 decoys so the lowest score of the more decoy case can only be lower or equal to that of the original $n = 2^{-9}$ run.

Figure 5.80 c) shows that the values of $\%\Delta AveIt$ in the more decoy case do not change but in the more perturbation case the values of $\%\Delta AveIt$ increases to 0% which means $AveIt$ of the more perturbation case is equal to that of $n = 1$. The change in the more perturbation case is because the number of total iterations is now nearly equal to that of $n = 1$.

Lastly, Figure 5.80 d) shows that more decoys do not significantly improve or worsen the values of $\%\Delta e_n$ from those in the $n = 2^{-9}$ run without additional iterations but more perturbations worsen the values of $\%\Delta e_n$ such that they are about equal to those for $n = 1$. While adding more decoys does not change efficiency, increasing perturbations does yield worse efficiency. As the simulation progresses efficiency becomes worsened because the values of the score improvement between consecutive iterations are most likely larger at the beginning of the simulation than those toward the end of the simulation. Thus, this suggests that more perturbations should not be added to a simulation for better efficiency.

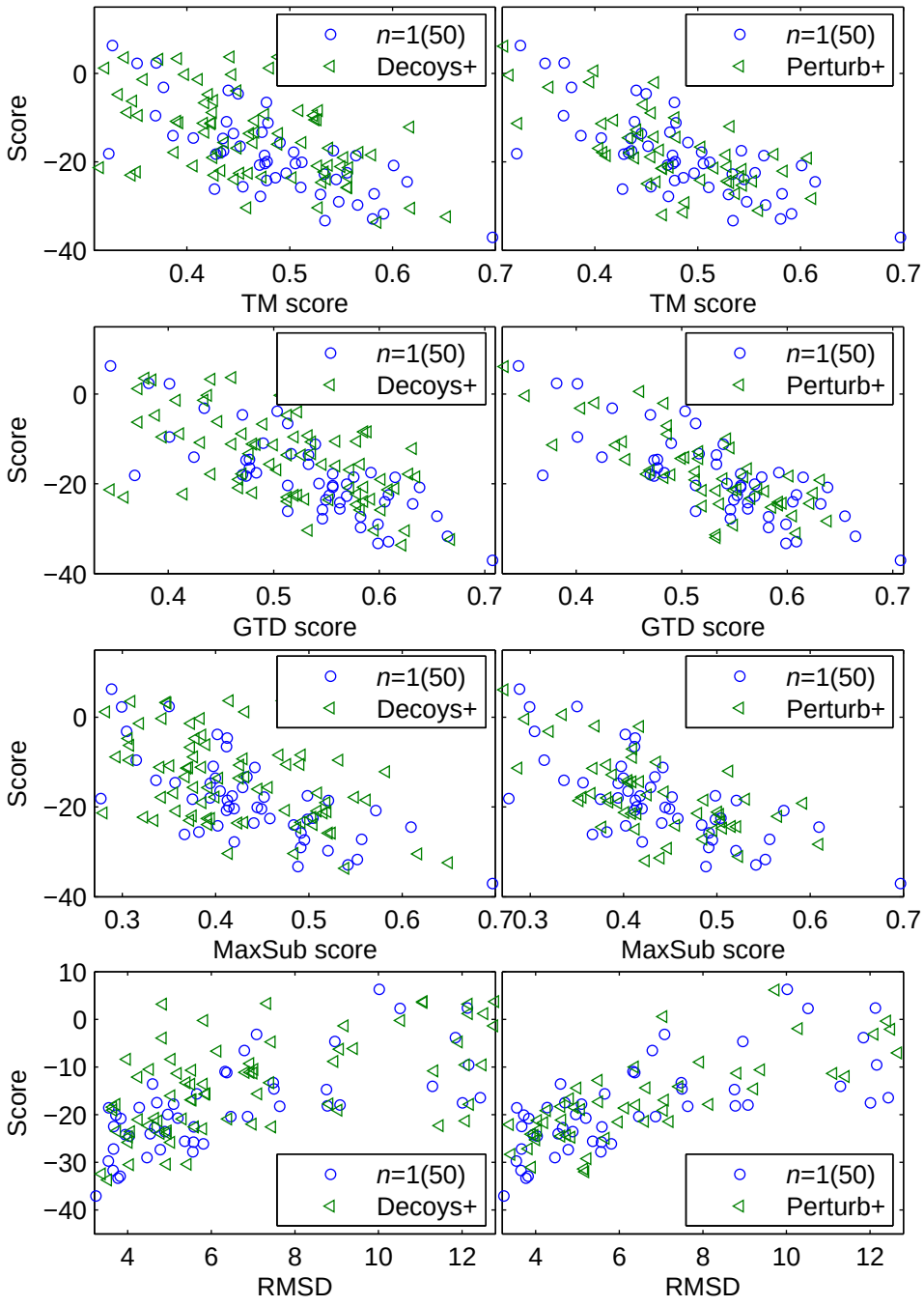

Figure 5.81: BFGS-XEL (*Hessian only*): The scores versus the TM, the GTD, the MaxSub scores and the RMSD values of resulting configurations for protein 1ubq. These quantities are from special runs with $n = 2^{-9}$ and either more decoys (Decoys+) or more perturbations (Perturb+). They are compared to those of $n = 1$ without varying the numbers of original decoys or perturbations. Lower score means better quality. The TM, the GTD, or the MaxSub score closer to 1 means a configuration more similar to the native conformation. RMSD value closer to 0 means a configuration more similar to the native conformation.

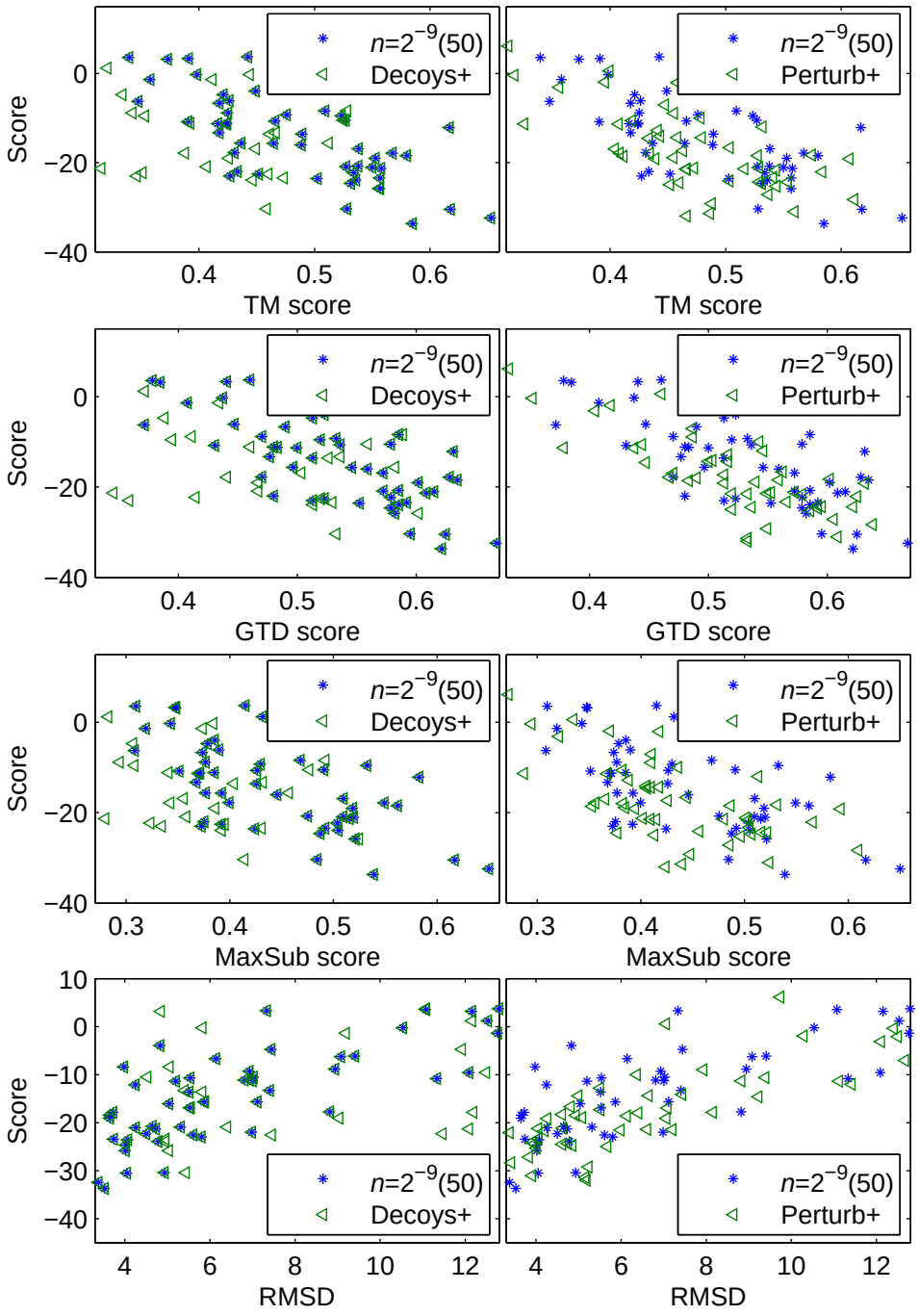

Figure 5.82: BFGS-XEL (*Hessian only*): The scores versus the TM, the GTD, the MaxSub scores and the RMSD values of resulting configurations for protein 1ubq. These quantities are from special runs with $n = 2^{-9}$ and either more decoys (Decoys+) or more perturbations (Perturb+). They are compared to those of $n = 2^{-9}$ without varying the numbers of original decoys or perturbations. Lower score means better quality. The TM, the GTD, or the MaxSub score closer to 1 means a configuration more similar to the native conformation. RMSD value closer to 0 means a configuration more similar to the native conformation.

For similarity evaluation the TM, the GTD, the MaxSub scores and the RMSD values of the resulting configurations from special runs with more decoys and more perturbations for protein 1ubq are evaluated. Figures 5.81 and 5.82 compare these quantities from special runs with $n = 2^{-9}$ and either more decoys (Decoys+) or more perturbations (Perturb+) to those of the $n = 1$ case and the $n = 2^{-9}$ respectively for which the original numbers of decoys and perturbations are not altered. Results from special runs with more decoys (Decoys+) and more perturbations (Perturb+) are shown as triangles, those from $n = 1$ are shown as circles, and those from $n = 2^{-9}$ are shown as stars. A TM, a GDT, or a MaxSub score closer to 1 means a configuration more similar to the native conformation. An RMSD value closer to 0 means a configuration more similar to the native conformation.

Figure 5.81 shows that with more decoys (Decoys+) the results are similar to those of $n = 1$ but cover slightly larger areas. With more perturbations (Perturb+) the results are also similar to those of $n = 1$. Figure 5.82 shows that the additional decoys do not have a high similarity and more perturbations do not improve or worsen the similarity in resulting configurations.

5.2.7.1 Conclusion of further analysis: same-iteration comparison

Overall adding more decoys does not change the quality, speed, or the efficiency of the BFGS-XEL (*Hessian only*) but increasing perturbations can improve quality in terms of the average score improvement and the lowest score while worsening speed and efficiency. Note that only one value of n and two proteins are used in this analysis. Running similar tests with other values of n and more proteins will increase the confidence of this conclusion.

5.2.8 Conclusion of XEL Algorithms with Probabilistic Search

The QNA-XEL, the QNA-XEL (*Hessian only*), the BFGS-XEL, and the BFGS-XEL (*Hessian only*) are implemented with the bracketing and Brent line search in an

optimization with a probabilistic search under the Rosetta platform. Simulations are run on 50 decoys from five proteins with 11 values of n including $n = 1$ which is the unmodified case. The results are evaluated in terms of quality, speed, and efficiency.

In terms of quality the average score improvement $AveSI$ and the lowest scores E_{lowest} are determined. Results show that $n > 1$ yields a lower $AveSI$, which means higher quality, than $n < 1$ and sometimes than $n = 1$. This is in agreement with a conclusion from Section 5.1 and Guideline *iv.* in Table 4.1. A larger n has smaller steps so the matrix estimation becomes more accurate which results in better quality. However, when n is too large, the quality becomes unexpectedly worse. This is because with steps too small a relative change in the score becomes so small that it satisfies a stopping criterion before converging to a solution and results in a worse quality.

Results also show that on average $n \neq 1$ yields a higher E_{lowest} than $n = 1$ but some cases of $n \neq 1$ can yield a lower E_{lowest} than $n = 1$. A lower E_{lowest} implies higher quality and global search performance. The results are in agreement with those from Section 5.1. This suggests that a lower E_{lowest} can be achieved, but simply having a constant exponent n for the duration of the optimization may not allow a lower value of E_{lowest} to be achieved. The adaptive exponential energy landscaping (AXEL) that changes n during the course of simulation is presented in Section 5.3.

For speed and efficiency evaluation the average iteration $AveIt$ and the efficiency e_n are determined. Results show that $n < 1$ yields a lower $AveIt$ and a lower e_n than $n > 1$ which implies better speed and efficiency. This however does not agree with the results seen in Section 5.1 which show that with the bracketing and Brent line search $n > 1$ has slightly better speed than $n < 1$. This inconsistency may be because the simulated simpler energy landscapes do not represent the score function of a protein molecule.

$n < 1$ yields better speed because the steps become larger so $n < 1$ paths travel

Table 5.5: The range of the percent difference of the average iteration $\%\Delta AveIt$ yielded by different algorithms with different values of n. Lower values imply better speed. The ranges are displayed from worse to better speed. Negative value means better speed than $n = 1$ and positive value means worse speed.

n	**QNA-XEL** (%)	**QNA-XEL** (*Hessian only*) (%)	**BFGS-XEL** (%)	**BFGS-XEL** (*Hessian only*) (%)	**Overall** (%)
2^{-9}	-29 to -42	-34 to -47	-24 to -41	-24 to -40	-24 to -47
2^{-7}	-28 to -41	-34 to -44	-20 to -37	-21 to -36	-20 to -44
2^{-5}	-25 to -36	-32 to -42	-18 to -33	-15 to -28	-15 to -42
2^{-3}	-21 to -28	-27 to -34	-8 to -27	-12 to -22	-8 to -34
2^{-1}	-8 to -15	-14 to -19	-2 to -11	-2 to -11	-2 to -19
2	3 to -3	8 to -1	9 to -3	7 to -2	9 to -3
4	0 to -10	7 to -6	6 to -10	3 to -10	7 to -10
6	-3 to -14	3 to -8	1 to -17	2 to -18	3 to -18
8	-12 to -21	-7 to -15	-6 to -24	1 to -17	1 to -24
10	-18 to -28	-9 to -22	-15 to -32	1 to -20	1 to -32

to the minima faster. However, this reasoning does not explain why $n > 1$ yields better speed than $n = 1$. Keep in mind that $n > 1$ steps may be shorter but it does not imply that the number of iterations will increase. Since the matrix estimation is more accurate with $n > 1$, the path converges to the closest solution which means the path travels a much shorter distance so it does not require as many steps. Also with steps too small a relative change in the score becomes so small that a stopping criterion is satisfied before converging to a solution. Therefore, $n > 1$ yields better speed than $n = 1$.

Comparing results from the XEL and the XEL (*Hessian only*) shows that different magnitudes of the steps yield similar trends but some deviations can be seen. Although the term β^{-1} in the XEL (*Hessian only*) gives significantly different magnitudes, the results are similar because the bracketing and Brent line search reduces their effect.

Table 5.6: The range of the percent difference of the average score improvement $\%\Delta AveSI$ yielded by different algorithms with different values of n. Lower values imply higher quality. The ranges are displayed from lower to higher quality. Negative values mean higher quality than $n = 1$ and positive values mean lower quality.

n	**QNA-XEL** (%)	**QNA-XEL** (*Hessian only*) (%)	**BFGS-XEL** (%)	**BFGS-XEL** (*Hessian only*) (%)	**Overall** (%)
2^{-9}	15 to 7	11 to 7	13 to 6	13 to 4	15 to 4
2^{-7}	11 to 7	12 to 7	11 to 7	9 to 4	12 to 4
2^{-5}	13 to 6	10 to 6	13 to 4	12 to 4	13 to 4
2^{-3}	13 to 4	12 to 5	11 to 2	7 to 1	13 to 1
2^{-1}	10 to 5	8 to 3	10 to 1	6 to -1	10 to -1
2	4 to -3	1 to -4	-1 to -10	-1 to -4	4 to -10
4	4 to 0	-2 to -12	4 to -10	2 to -4	4 to -12
6	6 to -5	-5 to -12	1 to -9	4 to -1	6 to -12
8	5 to 0	-3 to -9	3 to -5	5 to -1	5 to -9
10	8 to 0	-1 to -10	8 to 1	4 to -2	8 to -10

Results shows that when the XEL is implemented a trade-off between quality and speed must be considered. Tables 5.5 and 5.6 are look-up tables for the range of the percent difference of the average iteration and the average score improvement yielded by different algorithms with different values of n. Table 5.5 shows that when the QNA-XEL is implemented $n = 2^{-9}$ yields the best speed which is 29% to 42% higher than the unmodified case; however, Table 5.6 shows that although an improvement in speed is achieved quality may be reduced by 7% to 15%. If the BFGS-XEL is implemented and higher quality is desired, Table 5.6 suggests that $n = 2$ should be used because it gives 1% to 10% higher quality, but the trade-off in speed may be up to 9% as shown in Table 5.5.

Results show that XEL significantly improves the efficiency of the analytical minimization, especially with the values of n in the 2^{-9}–2^{-5} range. Figures 5.70 and 5.71 show that efficiency in the QNA-XEL and the QNA-XEL (*Hessian only*) can be

Table 5.7: The average TM, GDT, MaxSub scores, and RMSD values for XEL with various values of n and two special runs with more decoys or perturbations. All results are for protein 1ubq and from BFGS-XEL (*Hessian only*). The two special runs use $n = 2^{-9}$. A TM, a GDT, or a MaxSub score closer to 1 means an average configuration more similar to the native conformation. An RMSD value closer to 0 means an average configuration more similar to the native conformation. The values corresponding to the best results are in boldface.

n	**Score**	**TM score**	**GTD score**	**MaxSub score**	**RMSD**
2^{-9}	-14.3	0.482	0.530	0.441	6.60
2^{-7}	-15.5	0.487	0.535	0.448	6.51
2^{-5}	-14.7	0.482	0.532	0.443	6.60
2^{-3}	-16.9	0.479	0.526	0.439	6.57
2^{-1}	-18.5	**0.490**	0.535	**0.454**	6.49
1	-18.5	0.482	0.533	0.437	**6.41**
2	**-18.7**	0.478	0.528	0.439	6.58
4	-18.4	0.481	0.532	0.440	6.56
6	-17.1	0.476	0.527	0.435	6.54
8	-16.9	0.482	0.532	0.441	6.52
10	-17.2	**0.490**	**0.536**	**0.454**	6.49
More decoys	-14.1	0.464	0.514	0.423	7.14
More perturbations	-17.3	0.472	0.523	0.431	6.57

improved by 25% to 51% and 32% to 75%. Figures 5.74 and 5.75 show that efficiency in the BFGS-XEL and the BFGS-XEL (*Hessian only*) can be improved by 13% to 49% and 13% to 52%.

Another measure of quality is the similarity between resulting configurations and the native conformation. The TM, the GTD, the MaxSub scores and the RMSD values of resulting configurations for protein 1ubq are evaluated. Results show that XEL does not improve or worsen the similarity to the native conformation in terms of the TM, the GDT, the MaxSub scores, and the RMSD values. Table 5.7 presents average TM, GDT, MaxSub scores, and RMSD values for XEL with various values

Table 5.8: The percent difference of average TM, GDT, MaxSub scores, and RMSD values compared to $n = 1$ for XEL with various values of n and two special runs with more decoys or perturbations. All results are for protein 1ubq and from BFGS-XEL (*Hessian only*). The two special runs use $n = 2^{-9}$. A negative value means better performance than $n = 1$.

n	**Score (%)**	**TM score (%)**	**GTD score (%)**	**MaxSub score (%)**	**RMSD (%)**
2^{-9}	22.7	0	0.56	-0.92	2.96
2^{-7}	16.2	-1.04	-0.38	-2.52	1.56
2^{-5}	20.5	0	0.19	-1.37	2.96
2^{-3}	8.65	0.62	1.31	-0.46	2.50
2^{-1}	0	-1.66	-0.38	-3.89	1.25
2	-1.08	0.83	0.94	-0.46	2.65
4	0.54	0.21	0.19	-0.69	2.34
6	7.57	1.24	1.13	0.46	2.03
8	8.65	0	0.19	-0.92	1.72
10	7.03	-1.66	-0.56	-3.89	1.25
More decoys	23.78	3.73	3.56	3.20	11.39
More perturbations	6.49	2.07	1.88	1.37	2.50

of n and two special runs with more decoys or perturbations as described in Section 5.2.7. Table 5.8 presents the percent difference of these values compared to $n = 1$. These values are calculated such that a negative value means better performance than $n = 1$.

Table 5.7 shows that $n = 1$ yields a better average score and an average RMSD value than the other runs, but it yields similar and sometimes lower average TM, GDT, and MaxSub scores than other runs. Table 5.8 shows that average TM, GDT, and MaxSub scores for $n \neq 1$, including the special runs, are within 4% of those for $n = 1$ and 90% of these quantities are within 2%. This means for $n \neq 1$ the configurations are similar to those for $n = 1$ with fewer iterations. Note that although these results give an insight about the similarity of the resulting configurations, results

from many more decoys will increase the confidence in this conclusion because of the probabilistic nature of the process.

While results show that $n < 1$ can yield up to 47% better speed than $n = 1$, it can yield up to 15% lower quality in terms of the average score improvement. The benefits from better speed however may exceed this drawback and may be more desirable. In a practical protein prediction, running refinements on more decoys is more preferable than running longer refinements the same decoys because it means the higher chance of locating the global minimum energy conformation. Moreover, since the scoring functions are far from perfect [57, 76], most of the successful resulting conformations are merely close to the native conformations, and sometimes the lowest energy results from simulations are discarded because they contain inaccurate or incorrect folds [11]. Since XEL has shown almost no effect on the similarity of resulting configurations, XEL can yield configurations as similar to the native conformation as the unmodified case with less conformation cost. This means that XEL can significantly improve the protein prediction simulation even though it yields less average score improvement.

Results from same-iteration comparison show that when $n = 2^{-9}$ increasing the number of decoys does not lower the average score improvement or the lowest score and suggest that increasing the perturbations is not desirable. While this implies that simply increasing the number of decoys does not improve the quality, the confidence in this conclusion is low because only one case of n and two proteins are investigated. A study on more proteins and with more cases of n is needed for a more definite conclusion.

To improve quality without sacrificing speed, the AXEL presented in Section 5.3 is investigated.

5.3 BFGS with AXEL

Since when $n < 1$ the landscape is flattened and when $n > 1$ the global minimum becomes more prominent, different values of the exponent n yield different effects on the performance of the XEL. This is reflected on results from Section 5.2 which show that $n > 1$ yields better quality than $n < 1$ but worse speed and efficiency. Therefore, the energy landscape should be dynamically modified as the optimization search progresses.

To improve the overall performance the AXEL is composed of two parts, the AXEL schemes that describe how n changes during the course of conformation prediction, and the XEL or the varying-n XEL algorithms that are used in minimization. The AXEL schemes are categorized by when the exponent n changes as described in Section 4.3.2. The AXEL is investigated with several AXEL schemes on the BFGS-XEL (*Hessian only*) and the varying-n BFGS-XEL algorithm. Sections 5.3.1 and 5.3.2 present AXEL results from global and local AXEL schemes, respectively. Lastly, Section 5.3.3 concludes the BFGS with AXEL section.

5.3.1 Global schemes

Global AXEL schemes change the exponent n depending on the global location of the probabilistic search but n's are constant during each minimization portion of the simulation. Referring to Figure 5.43 that presents the refinement protocol of Rosetta, the global AXEL schemes change n before the minimization that allows all dihedral angles on the entire chain to change, or the minimization that allows only perturbed dihedral angles to change, or both. Since n is constant during the minimization, XEL algorithms are used with the global AXEL schemes. In this section the following global AXEL schemes are investigated with the BFGS-XEL (*Hessian only*):

Scheme A) This scheme is used as the control case for the global scheme. Two cases of Scheme A are implemented. First, n is randomly modified six times at points

before the minimization that allows all dihedral angles on the entire chain to change. Second, n is randomly modified at points before the minimization that allows only perturbed dihedral angles to change. The results are presented in Section 5.3.1.1.

Scheme B) The values of n to be used during the beginning, the middle, and the end of the simulation are determined from the linear regressions of the negative natural log of the absolute running average efficiency and the natural log of *unnormalized* iterations. The results are presented in Section 5.3.1.2.

Scheme C) The values of n to be used during the beginning, the middle, and the end of the simulation are determined from the linear regressions of the negative natural log of the absolute running average efficiency and the natural log of *normalized* iterations. Results presented in Section 5.3.1.2.

Scheme D) $n \neq 1$ is used during the first half of the simulation. Then $n = 1$ is used during the second half of the simulation and the convergence criterion is reduced by half. The results are presented in Section 5.3.1.3.

5.3.1.1 Scheme A: Random n

In this section two cases of Scheme A that randomly changes n are implemented. In Case (i) n is randomly modified six times at points before the minimization that allows all dihedral angles on the entire chain to change. In Case (ii) n is randomly modified at points before the minimization that allows only perturbed dihedral angles to change. Figure 5.83 presents a) the percent difference of average score improvement ($\%\Delta AveSI$), b) the percent difference of the lowest score ($\%\Delta E_{lowest}$), c) the percent difference of the average iteration ($\%\Delta AveIt$), and d) the percent difference of the efficiency ($\%\Delta e_n$) from the AXEL with Scheme A. Lower values mean better performance and negative values mean better performance than $n = 1$.

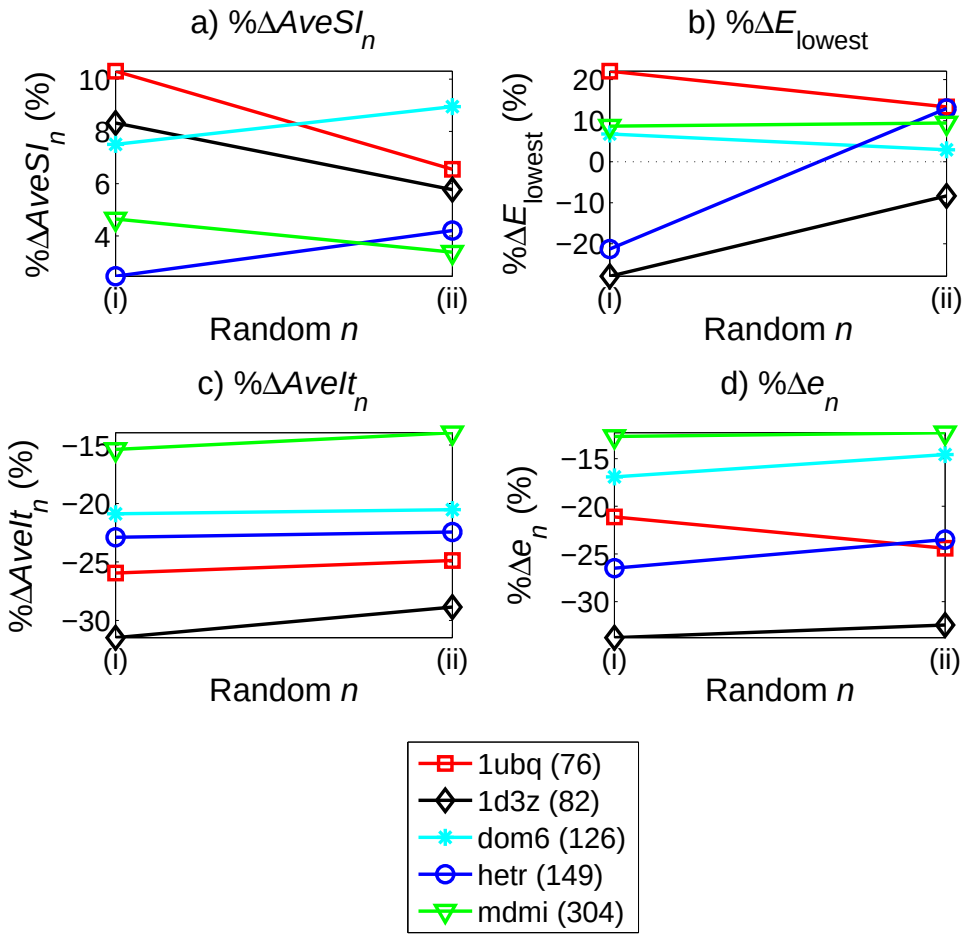

Figure 5.83: Scheme A: a) the percent difference of average score improvement $\%\Delta AveSI$, b) the percent difference of lowest score $\%\Delta E_{lowest}$, c) the percent difference of average iteration $\%\Delta AveIt$, and d) the percent difference of the efficiency $\%\Delta e_n$. Case (i) has n randomly modified at points before the minimization that allows all dihedral angles on the entire chain to change and Case (ii) has n randomly modified at points before the minimization that allows only perturbed dihedral angles to change. Lower values mean better performance and negative values mean better performance than $n = 1$.

Results (Figure 5.83 a) show that there is no relationship between the rate of the varying n and the improvement or worsening of the values of $\%\Delta AveSI$ compared those previously seen in Section 5.2. The values of $\%\Delta E_{lowest}$ (Figure 5.83 b) are also within the same range as those previously seen but in protein 1d3z Scheme A is able to yield 10% lower $\%\Delta E_{lowest}$ by changing n only a few times. The values of $\%\Delta AveIt$ (Figure 5.83 c) are also within the same range as those previously seen but higher than the best case of those previously seen. The values of $\%\Delta e_n$ (Figure 5.83 d) are also within the same range as those previously seen but generally have $\%\Delta e_n$ values higher. Since the results yield about the same values of $\%\Delta AveSI$ and $\%\Delta E_{lowest}$ but lower values of $\%\Delta AveIt$ and $\%\Delta e_n$ compared to those previously seen, this suggests that changing n blindly can worsen speed and efficiency without any improvement in quality.

5.3.1.2 Schemes B and C: Running average efficiency

To investigate if efficiency can be further improved, Schemes B and C are developed. The values of n to be used in these schemes are the values that give the best running average efficiency, defined below, during the beginning, the middle, and the end of the simulation. Schemes B and C determine n from the linear regressions of the negative natural log of the absolute running average efficiency ($-\ln|e_{\text{running}}|$ defined below) versus the natural log of *unnormalized* iterations and versus the natural log of *normalized* iterations, respectively.

The efficiency e_n in Equation (5.7) is the overall efficiency computed at the end of the simulation. To understand the effect of n on the performance during the different periods of simulations, the running average efficiency $e_{running}$ is evaluated.

$$(e_{\text{running}})_{j,i} = \frac{(\min([E_{1,i} \ldots E_{j,i}]) - E_{0,i})}{m_{j,i}/1000}; \quad i = 1 \ldots 50. \tag{5.9}$$

where $(e_{\text{running}})_{j,i}$ is the running average efficiency at the j^{th} configuration of the i^{th} decoy, $E_{1,i}$ is the score of the first configuration (input) of the i^{th} decoy, $E_{j,i}$ is the

Figure 5.84: The running average efficiency, $e_{running}$, versus iterations for protein 1d3z with various values of n. Lower values mean better running average efficiency.

score of the j^{th} configuration of the i^{th} decoy, and $m_{j,i}$ is the number of iterations of the j^{th} configuration of the i^{th} decoy. Then the average set of running average efficiency e_{running} for all 50 decoys is found. Since the set of running average efficiency for each decoy is not available for every iteration and it corresponds to a different set of iterations for different decoy, cubic spline data interpolation is used to obtain the running average efficiency from the same iteration set so that an average can be calculated.

Illustrating a relationship between e_{running} and iteration for various values of n of a representative example, 1d3z protein, Figure 5.84 shows that all n cases give almost identical logarithmic relationships. When these relationships are plotted in log scales, they become linear as shown in Figure 5.85 which illustrates that n that yields the best values of e_{running} has its graph lower than the others. Identifying the best values of n at the beginning, the middle, or the end of the simulation directly from these graph can be difficult because they are not perfect lines. Therefore, the

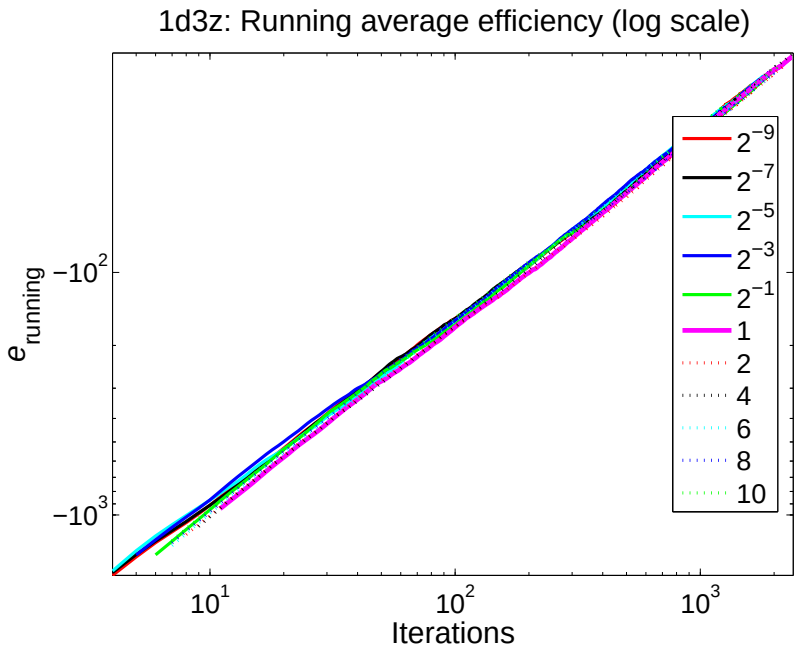

Figure 5.85: The running average efficiency, $e_{running}$, (log scale) versus iterations (log scale) with various values of n for protein 1d3z. Lower values mean better running average efficiency.

best values of n are determined from the linear regressions of these graphs. Two sets of the regressions which look similar to Figure 5.85 give n values for Schemes B and C.

Scheme B determines the values of n that give the best e_{running} from the linear regressions between $-\ln|e_{\text{running}}|$ and the natural log of iterations during the beginning, the middle, and the end of simulations for protein 1ubq, 1d3z, dom6, hetr, and mdmi. Scheme C determines values of n that give the best e_{running} from the linear regressions between $-\ln|e_{\text{running}}|$ and the natural log of *normalized* iterations. With unnormalized iterations in Scheme B the beginning, the middle, and the end periods of each simulation are defined by certain numbers of iterations, but with the normalized iterations in Scheme C the periods are defined by the length of the simulation itself. A chosen value of n yields the lowest regression line during each period which means the best performance.

Table 5.9: The values of n for Schemes B and C which give the best efficiency during the beginning, the middle, and the end of simulation periods for each protein from unnormalized and normalized iteration plots.

Period	Scheme	1ubq	1d3z	dom6	hetr	mdvi
Beginning	B	2	1	2	2	2
	C	2^{-9}	2^{-9}	2^{-9}	2^{-5}	2^{-9}
Middle	B	2	2	1	2	2
	C	2^{-9}	2^{-7}	2^{-9}	2^{-9}	2^{-9}
End	B	6	8	1	2^{-3}	2^{-5}
	C	2^{-9}	2^{-7}	2^{-9}	2^{-9}	2^{-9}

The values of n of Schemes B and C that give the best e_{running} during the beginning, the middle, and the end of the simulations for each protein are displayed in Table 5.9. $n = 2$ is often chosen for Scheme B and $n = 2^{-9}$ is often chosen for Scheme C. These values of n are used during their corresponding periods in the BFGS-XEL (*Hessian only*). Figure 5.86 gives a) the percent difference of average score improvement ($\%\Delta AveSI$), b) the percent difference of the lowest score ($\%\Delta E_{lowest}$), c) the percent difference of the average iteration ($\%\Delta AveIt$), d) and the percent difference of the efficiency ($\%\Delta e_n$) from the AXEL with Schemes B and C, respectively.

Figures 5.86 a) and b) show that Scheme B yields similar performance as $n = 1$ cases in terms of quality but Scheme C yields lower quality. Figure 5.86 a) shows that $AveSI$'s from Scheme B are comparable to those from $n = 1$ cases (within 0 to -3%), but those from Scheme C are 4 to 15% higher. This means obtaining the values of n from the normalized iteration data does not improve the performance of the algorithm regarding to the $AveSI$ values. Figure 5.86 b) shows that most lowest scores from Scheme B are comparable to those from $n = 1$ cases (within 0 to 5%) but for protein hetr Scheme B gives 15% lower lowest score. Scheme C gives a higher lowest score than Scheme B in small proteins but up to a 12% lower lowest score for larger proteins.

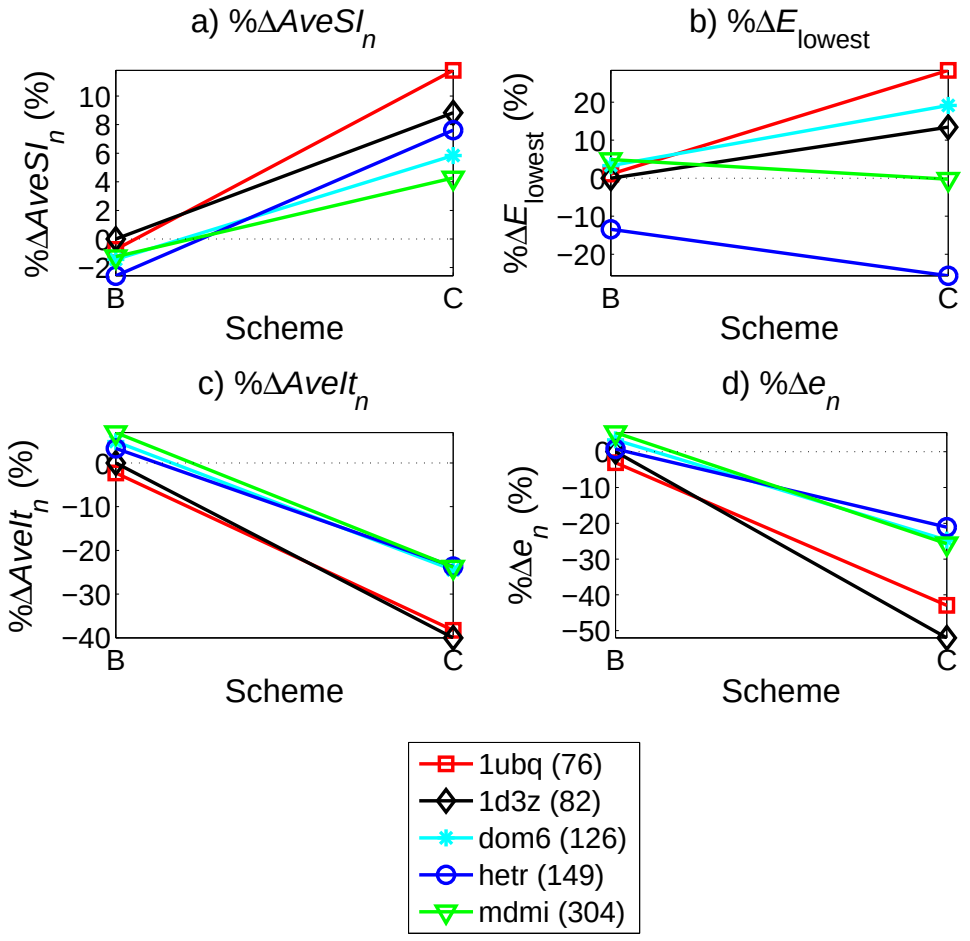

Figure 5.86: Schemes B and C: a) the percent difference of average score improvement $\%\Delta AveSI$, b) the percent difference of lowest score $\%\Delta E_{lowest}$, c) the percent difference of average iteration $\%\Delta AveIt$, and d) the percent difference of the efficiency $\%\Delta e_n$. Lower values mean better performance and negative values mean better performance than $n = 1$.

Figures 5.86 c) and d) demonstrate that Scheme B yields the same performance as $n = 1$ case in terms of speed and efficiency but Scheme C yields better speed and efficiency. In Scheme C, obtaining n's from the normalized iteration data yields the best speed and efficiency because most obtained n's are equal to 2^{-9}, which is the case that yields the best speed and efficiency in BFGS-XEL. In fact, three proteins, 1ubq, dom6, and mdmi have $n = 2^{-9}$ during the entire simulations. In Scheme B, obtaining n's from the unnormalized iteration data does not yield as good speed and efficiency because most obtained n's are 2, which yields about the same speed and efficiency as the $n = 1$ case.

The results from Schemes B and C show that determining the best n for each simulation period from the running average efficiency may not be suitable for the AXEL. Since the running average efficiency are cumulative, the n that gives the best result at the current simulation period is most likely the same as the best n at the last period. Having repeated n's yields almost the same results as the BFGS-XEL, in which n is constant.

5.3.1.3 Scheme D: Stricter convergence criterion

Scheme D changes both n and the convergence criterion. During the first half of the simulation $n \neq 1$ is used, and during the second half of the simulation n changes to 1 and the convergence criterion is reduced by half. Two values of $n \neq 1$ are used: $n = 2^{-9}$ and $n = 10$. Since the XEL algorithm yields higher efficiency than the unmodified case, this scheme is investigated to determine if quality can be improved by reducing speed with a stricter convergence criterion.

Figure 5.87 presents a) the percent difference of the average score improvement ($\%\Delta AveSI$), b) the percent difference of the lowest score ($\%\Delta E_{lowest}$), c) the percent difference of the average iteration ($\%\Delta AveIt$), and d) the percent difference of the efficiency ($\%\Delta e_n$), respectively. Lower values mean better performance and negative

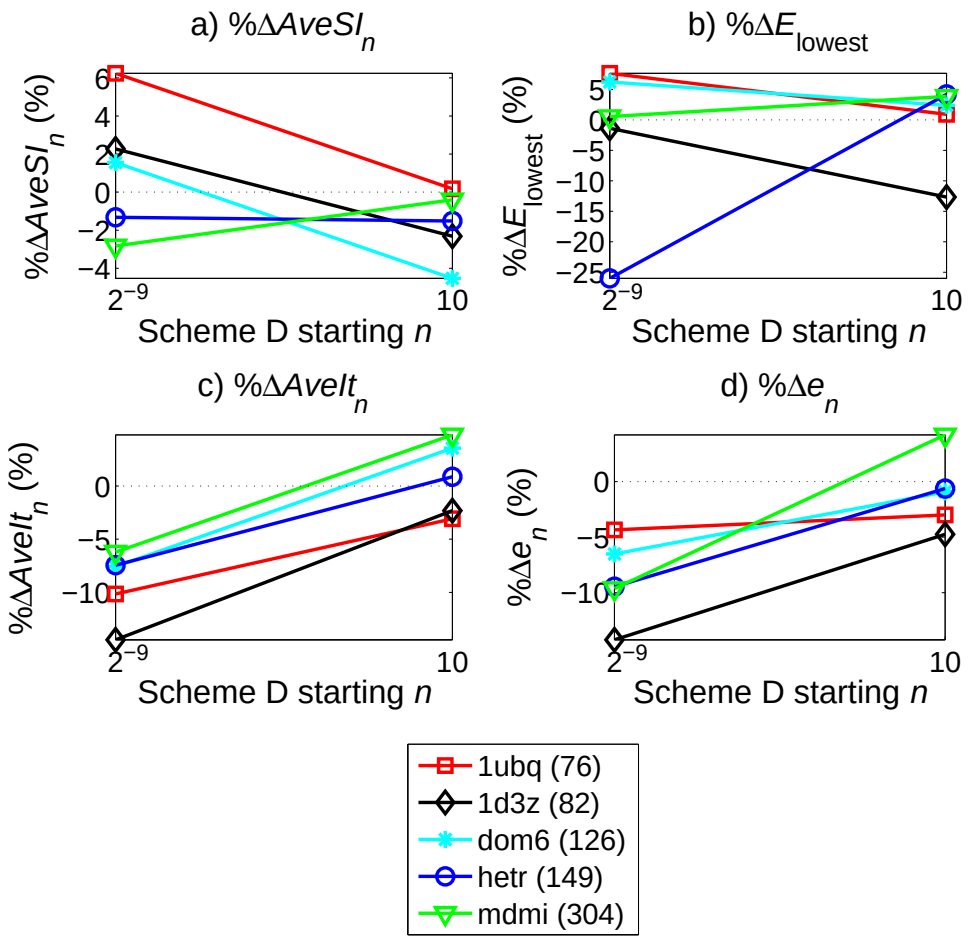

Figure 5.87: Scheme D: a) the percent difference of average score improvement $\%\Delta AveSI$, b) the percent difference of lowest score $\%\Delta E_{lowest}$, c) the percent difference of average iteration $\%\Delta AveIt$, and d) the percent difference of the efficiency $\%\Delta e_n$ from two values of starting n. Lower values mean better performance and negative values mean better performance than $n = 1$.

values mean better performance than $n = 1$.

Figures 5.87 a) and b) indicate that Scheme D performs as well as the $n = 1$ case in terms of quality. Figure 5.87 a) shows that almost all cases of Scheme D gives $AveSI$ within $\pm 5\%$ of $AveSI_{n=1}$ which is a significant improvement for the $n = 2^{-9}$ case compared to the constant $n = 2^{-9}$ case. Most cases with the starting $n = 10$ give lower $AveSI$ than cases with $n = 2^{-9}$, which yield up to 6% reduction in $\%\Delta AveSI$. For the $\%\Delta E_{lowest}$, Figure 5.87 b) shows that while only a few cases of Scheme D give slightly over 5% higher E_{lowest} than those of $n = 1$, most cases have E_{lowest} within

$\pm 5\%$ of $(E_{lowest})_{n=1}$ and other two cases have E_{lowest} 12.5% and 26% lower than those of $n = 1$.

Figures 5.87 c) and d) indicate that Scheme D performs better than the $n = 1$ case in terms of speed and efficiency. Figure 5.87 c) shows that most cases of Scheme D have lower numbers of iterations than $n = 1$. All cases with $n = 2^{-9}$ have lower $\%\Delta AveIt$ than cases with $n = 10$, but these numbers are higher than those of BFGS-XEL cases (Figure 5.67). This is due to the reduction of the convergence criterion during the second half of the simulation, which increases the number of iterations. Similar to $\%\Delta AveIt$, all $\%\Delta e_n$ for $n = 2^{-9}$ cases are lower than those of $n = 10$ cases, but these numbers are higher than those of of BFGS-XEL cases (Figure 5.75). The results of Scheme D demonstrate that the improvement of quality can be achieved by sacrificing speed and that the value of n at the beginning of the simulation have significant effect on the performance.

5.3.2 Local schemes

In local AXEL schemes, n changes depending on the information derived from the energy landscape which conditionally happens during the minimization. Since n varies during the minimization, the varying-n XEL algorithms are used with the local AXEL schemes. This section presents the results from simulations in which the varying-n BFGS-XEL is implemented. Several local AXEL schemes are investigated but only the following representative examples are presented in this section:

Scheme E) n changes such that the error residue is bounded. The results are presented in Section 5.3.2.1.

Scheme F) n changes when one of the criteria is met. Figure 5.89 gives the pseudo code for Scheme F). The results are presented in Section 5.3.2.2.

Note that a scheme that randomly changes n is not used as a control case for the local AXEL schemes because the changes can be too large and cause the minimization

to diverge.

5.3.2.1 Scheme E: Bounded error residue

Following Guideline *iii.* in Table 4.1 several schemes that change n such that the scalar multiplier λ (Equation 4.21) is bounded have been investigated, but they yield unsuccessful results and often diverge. Scheme E follows a slightly different approach by changing n such that the magnitude of the term $\lambda(n-1)E^{-1}$ is less than 1. As this term multiplies the first-order term of the XEL error residue ρ^* (Equation 4.27), a bounded $\lambda(n-1)E^{-1}$ implies a bounded $|\rho^*|$. Although ρ^* is not derived for the varying-n BFGS-XEL, by substituting $\hat{H}$ for K^{-1} in λ the modified $\lambda(n-1)E^{-1}$ are used. This is because the behaviors of the modified $\lambda(n-1)E^{-1}$ are assumed to be similar to the $\lambda(n-1)E^{-1}$ of the Newton-XEL.

Figure 5.88 presents a) the percent difference of the average score improvement ($\%\Delta AveSI$), b) the percent difference of the lowest score ($\%\Delta E_{lowest}$), c) the percent difference of the average iteration ($\%\Delta AveIt$), and d) the percent difference of the efficiency ($\%\Delta e_n$). In Case (i) there is no limit for the value of n and in Case (ii) only the value of n between 2^{-9} and 10 is allowed. Lower values mean better performance and negative values mean better performance than $n = 1$.

Figure 5.88 a) and b) shows that Scheme E slightly improves the quality and can yield higher quality in some proteins. As the values of $\%\Delta AveSI$ of hetr can be lower than -7%, the values of $\%\Delta E_{lowest}$ of hetr and 1d3z are -15% and 25%. On average the values of $\%\Delta AveSI$ and $\%\Delta E_{lowest}$ are slightly lower than those yielded by a constant n.

Figure 5.88 c) and d) shows that Scheme E only slightly worsens speed and efficiency compared to $n = 1$ as almost all values of $\%\Delta AveIt$ and $\%\Delta e_n$ are within the $\pm 5\%$ from zero. While the values of $\%\Delta AveIt$ (Figure 5.88 c) are about the same as those seen in the worst case of the constant n (Figure 5.66), the values of $\%\Delta e_n$ are

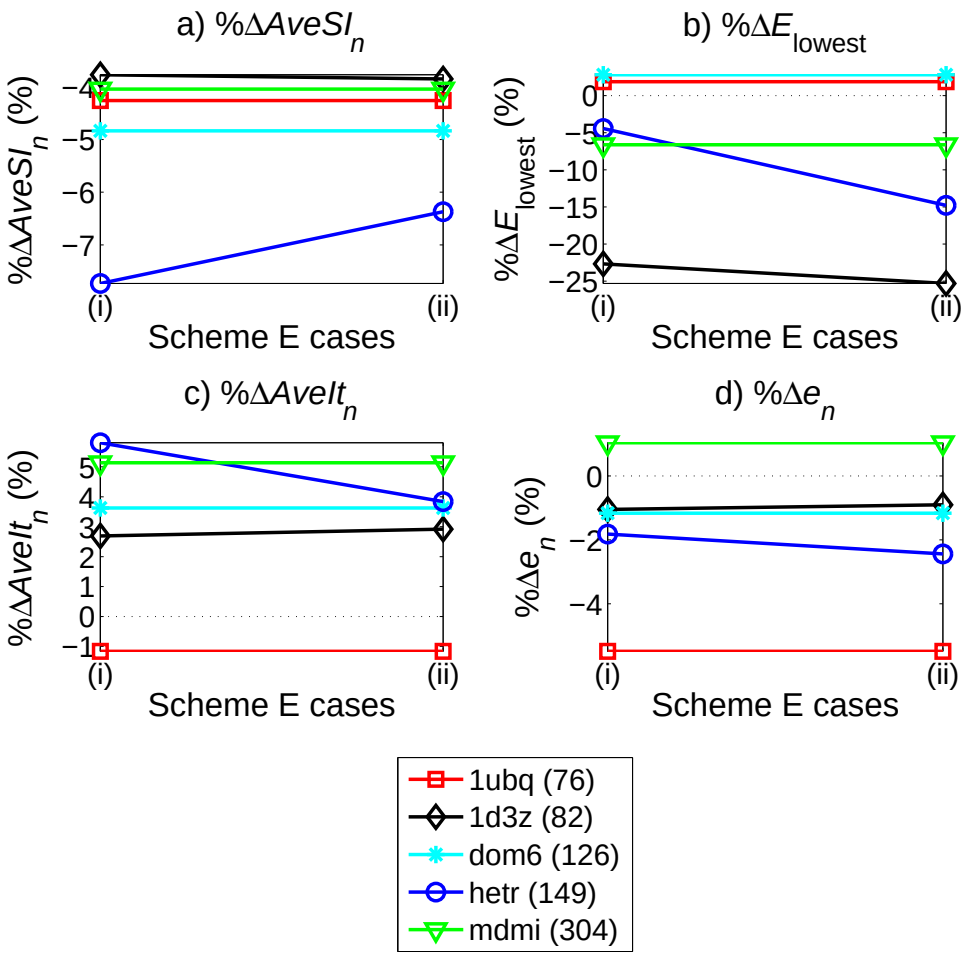

Figure 5.88: Schemes E: a) the percent difference of average score improvement $\%\Delta AveSI$, b) the percent difference of lowest score $\%\Delta E_{lowest}$, c) the percent difference of average iteration $\%\Delta AveIt$, and d) the percent difference of the efficiency $\%\Delta e_n$ from Schemes E cases. Case (i) has no limit for n and Case (ii) only allows the values of n to be between 2^{-9} and 10. Lower values mean better performance and negative values mean better performance than $n = 1$.

slightly lower than those seen in the worst case (Figure 5.74) because of lower values of $\%\Delta AveIt$.

The results show that on average Scheme E yields slightly higher quality for all proteins and worsens speed and efficiency no more than the worst case of the constant n cases.

5.3.2.2 Scheme F: Path and landscape characteristic criteria

Pseudo-code: Varying-n BFGS-XEL with Scheme F and a line search

Initialize ϵ, n_1, θ_0, θ_1, $\hat{H}_0^*$, and μ_i where $i = 1, \ldots, 6$
Calculate E_0 and τ_0
for $k = 1, \cdots, k_{max}$ **do**
  Calculate E_k, and τ_k
  if $\left| \dfrac{E_k - E_{k-1}}{E_{k-1}} \right| < \epsilon$ **then**
    `break` // Convergence criterion met
  end if
  Determine n_{k+1} and $\hat{h}_{n\,k}^*$
  if $k > \mu_1$ **then**
    // Change n only after μ_1^{th} iteration
    if $\dfrac{\tau_k \cdot \tau_{k-1}}{|\tau_k|\,|\tau_{k-1}|} > \mu_2$ **then**
      // An angle between the consecutive gradient is too large
      Increase n // Take a smaller step
    else if $\dfrac{|E_k - E_{k-1}|}{|E_{k-1}|} < \mu_3$ & $|\tau_k| > \mu_4$ **then**
      // Change in energy is small but the gradient is large
      Decrease n // Take a larger step
    else if $\dfrac{\hat{h}_{\theta\,k-1}^* \cdot \tau_k}{\left|\hat{h}_{\theta\,k-1}^*\right| |\tau_k|} < \mu_5$ **then**
      // The step is almost perpendicular to the gradient
      Increase n
    else if $\dfrac{k}{\mu_6} = 0$ **then**
      Decrease n // Increase step size every μ_6^{th} iteration
    end if
  end if
  Calculate $\hat{H}_k^*$
  $\tau_k^* = \beta_k^k \tau_k$
  $\hat{h}_{\theta\,k}^* = \hat{H}_k^* \left(1 + \left(\dfrac{1}{n_k} + \ln|E_k|\right) \hat{h}_{n\,k}^*\right) \tau_k^*$ // Varying-n BFGS-XEL step
  $\alpha_k^* =$ `line_search`$(\hat{h}_{\theta\,k}^*)$ // Determine the gain with a line search
  $\theta_{k+1} = \theta_k + \alpha_k^* \hat{h}_{\theta\,k}^*$ // $(k+1)^{\text{th}}$ solution
end for

Figure 5.89: A pseudo-code for Scheme F with varying n BFGS-XEL

Scheme F changes n in a systematic manner which is based on the observations from Section 5.1. Following Guideline *iv.* n is increased when the estimated modified Hessian inverse $\hat{H}^*$ is inaccurate. Therefore, $n > 1$ is used at the beginning of the simulation and n is changed only after the certain number of iterations. Three criteria are used to determine how and when to change n. (1) If the energy landscape changes too much, which occurs when the size of an angle between the current and the previous gradient exceeds a certain criterion, the step size is reduced by increasing n. (2) If the step size becomes too small, which occurs when a change in score is smaller than a criterion but a magnitude of the gradient is larger than a criterion parameter, the step size is increased by decreasing n. (3) When the $\hat{H}$ is not accurate, the step size is decreased by increasing n. $\hat{H}$ is considered inaccurate if the the step is almost perpendicular to the current gradient, therefore n is increased when an angle between the gradient and the step exceeds a criterion parameter. When none of the criteria are met, a larger step size is preferred for better speed, so n is decreased every certain number of iterations. Figure 5.89 gives a pseudo code for Scheme F). Two sets of criterion parameters i) and ii) shown in Table 5.10 are implemented and their results are presented in Figure 5.90. These parameter sets are those that yields the best results in preliminary tests implemented without the probabilistic search. Most parameters are kept the same as they do not significantly affect the results. Although Set ii) has some stricter criterion parameters, their results are similar.

Table 5.10: Two sets of criterion parameters for Scheme F.

Set	μ_1	μ_2	μ_3	μ_4	μ_5	μ_6
i)	3	0.707 (45°)	0.1 (10%)	0.5	0.259 (75°)	5
ii)	3	0.866 (30°)	0.1 (10%)	0.5	0.5 (60°)	5

Figure 5.90 presents a) the percent difference of the average score improvement ($\%\Delta AveSI$), b) the percent difference of the lowest score ($\%\Delta E_{lowest}$), c) the percent

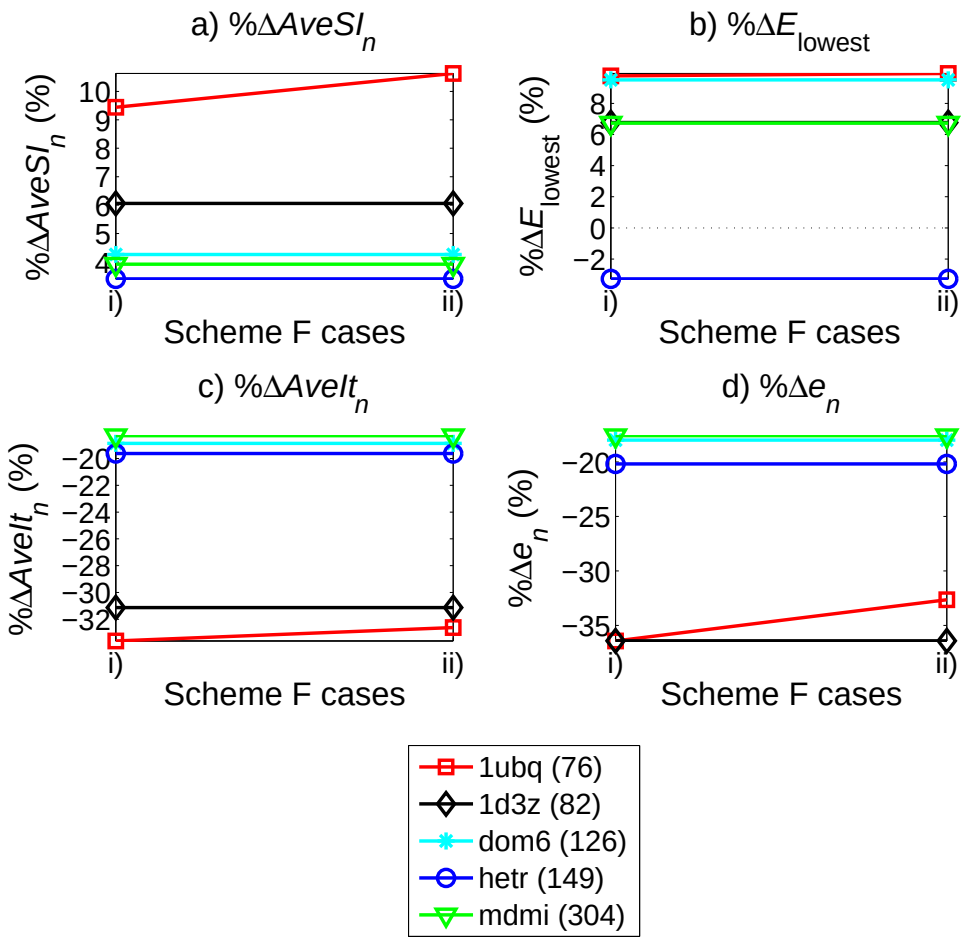

Figure 5.90: Scheme F: a) the percent difference of average score improvement $\%\Delta AveSI$, b) the percent difference of lowest score $\%\Delta E_{lowest}$, c) the percent difference of average iteration $\%\Delta AveIt$, and d) the percent difference of the efficiency $\%\Delta e_n$ from two sets of criterion paremeters. Lower values mean better performance and negative values mean better performance than $n = 1$.

difference of the average iteration ($\%\Delta AveIt$), and d) the percent difference of the efficiency ($\%\Delta e_n$). Lower values mean better performance and negative values mean better performance than $n = 1$.

Figure 5.90 a) and b) indicates that on average Scheme F yields lower quality than the $n = 1$ case. As proteins 1ubq and 1d3z yield more than 5% higher $AveSI_{n=1}$, almost all proteins yield 5% higher $\%\Delta E_{lowest}$. The achieved quality is similar to that seen in the constant $n < 1$ case (Figures 5.50 and 5.58).

Figure 5.90 c) and d) indicates that Scheme F yields significantly better speed

and efficiency than the $n = 1$ case. Both $\%\Delta AveIt$ and $\%\Delta e_n$ are dependent on the size of protein chains. That is they decrease as the size of protein chains increases. The achieved speed and efficiency are similar to that seen in the constant $n < 1$ case (Figures 5.66 and 5.74).

The results of Scheme F demonstrate that systematically changing n yields about the same results as the constant $n < 1$ case and two sets of the criterion parameters yields almost identical results. This is most likely because n is decreased more often than increased, stricter parameters μ_3 and μ_4 and larger μ_6 may yield different results. These parameters do not seem to affect the results in the preliminary test because the simulated energy landscapes most likely do not represent the score function of the protein molecule.

5.3.3 Conclusion of BFGS with AXEL

This section presents results of the AXEL applying to the BFGS algorithm. Several global and local AXEL schemes are implemented with the BFGS-XEL (*Hessian only*) or the varying-n BFGS-XEL and results are presented.

While the results from the global and local AXEL schemes do not give any insight into the global and the local effects of the AXEL on the prediction simulation, they show that the AXEL can improve quality or speed but rarely both at the same time. With Scheme A, results show that randomly changing n can worsen speed without improving quality. With Schemes B and C results show that choosing n for different period of the simulation from the running average efficiency is not suitable for the AXEL. Results from Scheme D show that higher quality can be achieved by sacrificing speed and that the value of n at the beginning of the simulation has a significant effect on the performance. The results show that Scheme E yields the best results in terms of quality than the other schemes as a lower $AveSI$ than that of the constant n case can be achieved in all proteins while the $AveIt$ is not more than 5% of the $n = 1$

case. Lastly, the results from Scheme F demonstrate that systematically changing n yields about the same results as the constant $n < 1$ case.

While most schemes yield the performance that can be achieved by constant n, Scheme E yields at least 4% lower $AveSI$ for all proteins which are not achieved by any constant n. Although only the varying-n BFGS-XEL is implemented with Scheme E, the better improvement seen here is expected to be found in other algorithms because of the consistency seen in results from cases with the constant n.

5.4 Conclusion of Simulation Results and Discussions

The effects of the XEL on the Newton's method, the QNA, and the BFGS method are investigated with two implementations. First, the investigated algorithms are implemented without a probabilistic search on a simulated two-dimensional energy landscape and second they are implemented with a probabilistic search on a multi-dimensional energy landscape. The AXEL is implemented with a probabilistic search on a multi-dimensional energy landscape.

Results from both implementations show that better quality or speed can be achieved by the XEL, but not both at the same time. Both results show that the quality can be achieved with $n > 1$ which is in agreement with Guideline *iv.* presented in Table 4.1. In terms of speed the results from the two implementations do not agree since $n > 1$ yields better speed in the first implementation but $n < 1$ yields better speed in the second implementation. In terms of the global search performance or ability to locate the global solution, on average the XEL yields higher lowest score than that of $n = 1$ case which implies worse global search performance.

The results from the XEL implementation with a probabilistic search show that a trade-off between the quality and speed must be considered when the XEL is implemented. Tables 5.5 and 5.6 on pages 213–214 are provided to be used as look-up tables. The results also show that XEL has no effect on the similarity of the resulting

configurations to the native conformation.

The results from the AXEL shows that quality or speed can be improved but rarely both at the same time. Scheme E yields at least 4% lower *AveSI* for all proteins compared to the XEL with constant n; however, the rest of AXEL yields roughly the same results. Results lead to three recommendations for better quality in terms of average score improvement, better speed in terms of the average iteration, and better efficiency.

To improve speed by 15% to 47%, the XEL with n within 2^{-9}–2^{-5} should be used. This range is suggested because of the consistent improvement of speed in all algorithms. Table 5.5 shows that for the QNA-XEL and the QNA-XEL (*Hessian only*) speed can be 25% to 42% and 32% to 47% better than the unmodified case and for the BFGS-XEL and the BFGS-XEL (*Hessian only*) speed can be 18% to 41% and 15% to 40% better. The XEL with n within 2^{-9}–2^{-5} is suggested for the first round of the refinement to evaluate a large set of decoys so that a better subset can be obtained for further refinement.

To improve quality by 4% to 7%, the AXEL with Scheme E should be used. Although only the varying-n BFGS-XEL were implemented, Scheme E is expected to yield similar improvement with the varying-n QNA-XEL. The varying-n XEL with Scheme E is suggested for the further round of the refinement for improved quality.

To improve efficiency by 13% to 75%, the XEL with n within 2^{-9}–2^{-5} should be used. Figures 5.70 and 5.71 show that the efficiency in the QNA-XEL and the QNA-XEL (*Hessian only*) cases can be 25% to 51% and 32% to 75% better than the unmodified case. Figures 5.74 and 5.75 show that the efficiency in the BFGS-XEL and the BFGS-XEL (*Hessian only*) cases can be 13% to 49% and 13% to 52% better than the unmodified case. The same range of n is recommended for improving speed.

CHAPTER VI

CONCLUDING REMARKS

This thesis develops and demonstrates the exponential energy landscaping (XEL) and the adaptive exponential energy landscaping (AXEL) for an analytical optimization algorithm in protein conformation prediction. The effects of XEL are investigated on Newton's method, the quasi-Newton algorithm (QNA), and the Broyden-Fletcher-Goldfarb-Shanno (BFGS) algorithm. The AXEL is a combination of a scheme for adaptively modifying energy landscape and the XEL algorithms.

XEL shares traits with hypersurface deformation [4, 5, 50] and the energy landscape flattening [87] in terms of modifying the energy landscape. However, the XEL has the following attributes that set it apart from the aforementioned techniques:

- XEL is applicable to any energy function.
- XEL is only applicable for an analytical optimization algorithm.
- XEL does not require remapping to the original surface.

This work addresses the bottleneck in protein conformation prediction which is the analytical optimization algorithm and provides a solution for improving speed and efficiency. The work also provides a potential solution for improving quality.

6.1 Contributions

This work makes the following contributions that are different from previous work:

1. **A theoretical basis for applying a method of the XEL to analytical optimization algorithms.**
    - The XEL that does not change the minima of the landscape and is applicable to any energy function is developed.

- The Newton's method, the QNA, and the BFGS algorithm with XEL are derived.
- Two methods of adaptive energy landscaping are derived: adaptive-n XEL algorithms and varying-n XEL algorithms with AXEL schemes.

2. **Analyses for theoretical validation and characterization of the XEL.**
   - The positive definite property of the Newton's method with XEL (Newton-XEL) is studied and conditions that give a positive definite Hessian matrix of the Newton-XEL are found.
   - The XEL is found to affect only the magnitude of the Newton step but not the direction. The scalar multiplier is found to be eigenvalue of a matrix and its properties are used to derive the error residue of the Newton-XEL.
   - The error residue of the Newton-XEL is found to be on the same order of that of Newton's method which implies that they have similar convergence property.
   - The scalar multiplier λ is also found to be inversely dependent of the exponent n.
   - The exponent n is found to weight the energy landscape information in relation to the unweighted Hessian matrix.
   - Guidelines for changing the exponent n in AXEL are developed.

3. **Simulation validation of the application and the effect of the XEL.**
   - XEL is found to be significantly affected by different line search algorithms.
   - XEL is found to significantly improve efficiency in the analytical portion of the protein prediction.
   - When XEL is implemented a trade-off between speed and quality must be considered.

- XEL is found to have little effect on the similarity of resulting configurations to the native conformation.
- AXEL is mostly found to yield similar performance as XEL.
- Recommendations for improving speed, quality, or efficiency are presented.

This work is complementary to [87]; both are applicable to any energy function and do not require remapping but [87] applies landscaping to the Monte Carlo method, a probabilistic optimization algorithm. Comparing this work to [87] can be done only by comparing the effects of landscaping on the optimization algorithm which was not studied in [87].

To put the above contributions into context, a brief summary is given here. Chapter 1 discusses that the bottleneck in protein conformation prediction is the analytical optimization algorithm and it should be addressed. Chapter 2 presents the literature review of several aspects in the protein conformation prediction problem. This leads to the objective of this research which is to develop a method of energy landscaping that enhances the efficiency of the analytical algorithms for the protein prediction problem. Chapter 3 presents the derivation and the application to the protein folding problem of the investigated algorithms which are Newton's method, the QNA, and the BFGS algorithm.

Chapter 4 presents the XEL, the AXEL, and their derivation on investigated optimization algorithms. The XEL modifies the energy landscape such that the minima stay at the same locations. The Newton's method, the QNA, and the BFGS algorithm are applied to the modified energy function and yield Newton-XEL, QNA-XEL, and BFGS-XEL algorithms. The Newton-XEL direction which determines the direction of the optimization step is proved to be parallel to the Newton direction. The convergence property of the Newton-XEL method is found to be similar to that of Newton's method as their error residues are on the same order. Several properties of the Newton-XEL yield guidelines for changing the exponent n of the XEL to achieve

certain operational characteristics such as improved stability or larger optimization step.

In the AXEL, the investigated algorithms are applied to the adaptively modified energy function, which is also of a function of n, and they yield adaptive-n XEL and varying-n XEL algorithms. Adaptive-n XEL algorithms simultaneously calculate the changes in n and the dihedral angles resulting in difficulties in reaching convergence. Varying-n XEL algorithms only calculate the change in dihedral angles based on the changes of n determined by the AXEL schemes. The AXEL schemes are categorized by when n changes as either global or local schemes.

Chapter 5 first presents and discusses the effects of the XEL on the Newton's method, the QNA, and the BFGS method on two implementations, with and without a probabilistic search. Then the AXEL implemented on the BFGS with a probabilistic search is presented and discussed. The results are evaluated in terms of quality, speed, and efficiency. While quality in terms of similarity are not affected by XEL, XEL can achieve better quality in terms of average score improvement or better speed in terms of average iteration. A better quality compared to unmodified case can be achieved with $n > 1$ and a better speed can be achieved with $n < 1$. However, the AXEL results show that both quality or speed can rarely be improved at the same time. This indicates that the trade-off between quality and speed must be considered when the XEL is implemented. The results lead to the following recommendations: (Note that quality is in terms of average score improvement and speed is in terms of the average iterations.)

- To improve *speed* by 15% to 47%, the XEL with n within 2^{-9}–2^{-5} should be used. Quality may be worsened by 4% to 15% on that range of n.
- To improve *quality* by 4% to 7%, the AXEL with Scheme E should be used. Speed may be worsened up to 6%.
- To improve *efficiency* by 13% to 75%, the XEL with n within 2^{-9}–2^{-5} should be

used.

Since in protein folding the conformation space is very large, sampling must be extensive to increase the possibility of covering the area that contains the global minimum. As a result, the process usually starts with several thousand decoys even for a small molecule. The first few rounds of the refinement are used to differentiate the good from the bad decoys. It is usually low resolution and does not required high quality. After the first or the second round of refinement, the best decoys are picked for the further round of refinement that usually requires higher quality. The XEL with n within 2^{-9}–2^{-5} is suggested for the first few rounds of the refinement and the AXEL with Scheme E is suggested for the higher refinement round.

While several investigated algorithms are derived and implemented, this work is not intended to compare the performance across different algorithms, but to compare the XEL algorithm to the unmodified case.

6.2 Future Work

Since the XEL is much more successful in increasing speed than quality, future work should be intermediately focused on further improving the quality and later focused on improving both simultaneously by developing an AXEL scheme. One approach can be the improvement or the variation of Scheme E. Another approach can be an improved global AXEL scheme as the current schemes yields the same performance as the constant n cases. Another study can combine the XEL to landscaping on the probabilistic optimization algorithm, such as energy landscape flattening, while another study can revisit the adaptive-n XEL approach by integrating a robust weighting algorithm.

With more computational resources at hand, a study of the global effect of the XEL can be done by implementing the XEL to a full-scale protein prediction process

which includes several rounds of full-atom refinement. Implementing XEL on the full-atom refinement will allow the effect of different energy functions to be studied. Since the improvement of the speed tends to be reduced for a larger protein, more proteins with higher residues will allow the effect of the protein size on the performance of XEL to be studied. Also the effect of different secondary structural contents in proteins on the performance of XEL can be studied. For example, proteins with large and small numbers of α-helix contents can be compared with those with large and small numbers of β-strand contents and with each other. While this work gives an insight about the similarity of the resulting configurations, future work should evaluate the similarity from the results of many more decoys and proteins. This will allow a conclusion to be drawn with more confidence because of the probabilistic nature of the process.

This work investigates the effects of the XEL on a few Newtonian algorithms. A study can be done to determine if other algorithms such as the Davidon-Fletcher-Powell (DFP) or the Broyden family can be improved by the XEL. Another study can compare the performance across these algorithms including the QNA and BFGS algorithm investigated here. Another study can investigate gradient-based optimization algorithms, such as the conjugate gradient method, that do not need Hessian calculation or estimation.

The energy landscape can be lowered or raised by a constant such that all energy values become negative or positive. A study can investigate the effects of lowering and raising the landscape on the performance of XEL in different algorithms.

A study of the application of the XEL on probabilistic methods can be investigated. An application of the XEL to the Monte Carlo method can possibly be done by modifying the energy term in the Boltzmann factor. With the exponent of $n < 1$ the Boltzmann factor becomes smaller which gives the same effect as flattening the energy landscape.

APPENDIX A

SUPPLEMENTARY DOCUMENTS

Appendix A provides supplementary data to Chapter 5, Section A.1.

A.1 QNA-XEL and BFGS-XEL Combining with Probabilistic Search

A.1.1 QNA-XEL and QNA-XEL (*Hessian only*)

Figures A.1 and A.2 present the percentage of decoys found by the QNA-XEL with lower, equal, or higher score or number of iterations than those of $n = 1$ for protein 1ubq, 1d3z, dom6, hetr, and mdmi. Figures A.3 and A.4 present results found by the QNA-XEL (*Hessian only*).

A.1.2 BFGS-XEL and BFGS-XEL (*Hessian only*)

Figures A.5 and A.6 present the percentage of decoys found by BFGS-XEL with lower, equal, or higher score or number of iterations compared to $n = 1$ for protein 1ubq, 1d3z, dom6, hetr, and mdmi. Figures A.7 and A.8 present results found by BFGS-XEL (*Hessian only*).

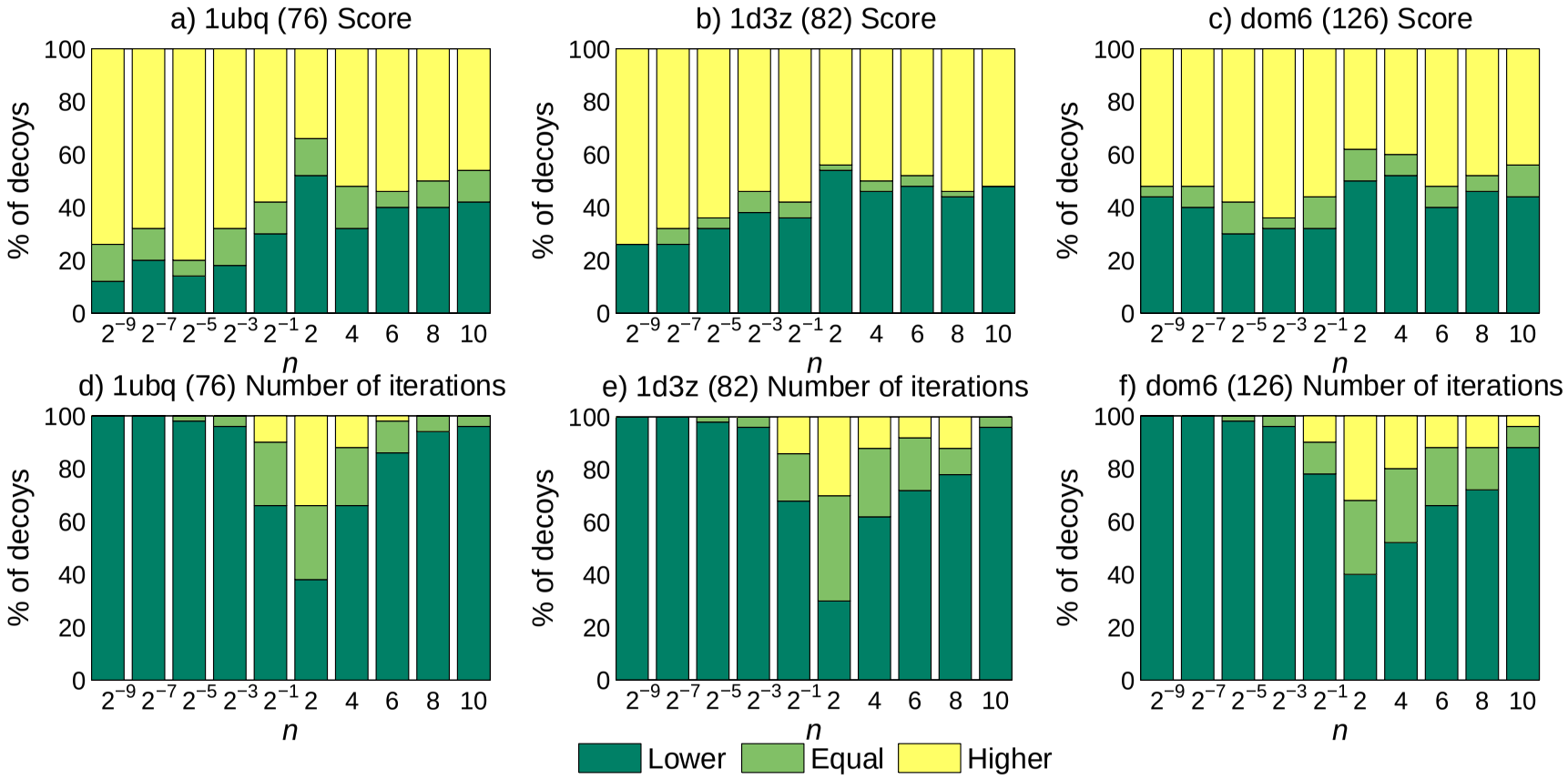

Figure A.1: QNA-XEL: The percentage of decoys with less, equal, higher scores or number of iterations compared to $n = 1$ for proteins 1ubq (a and d), 1d3z (b and e), and dom6 (c and f).

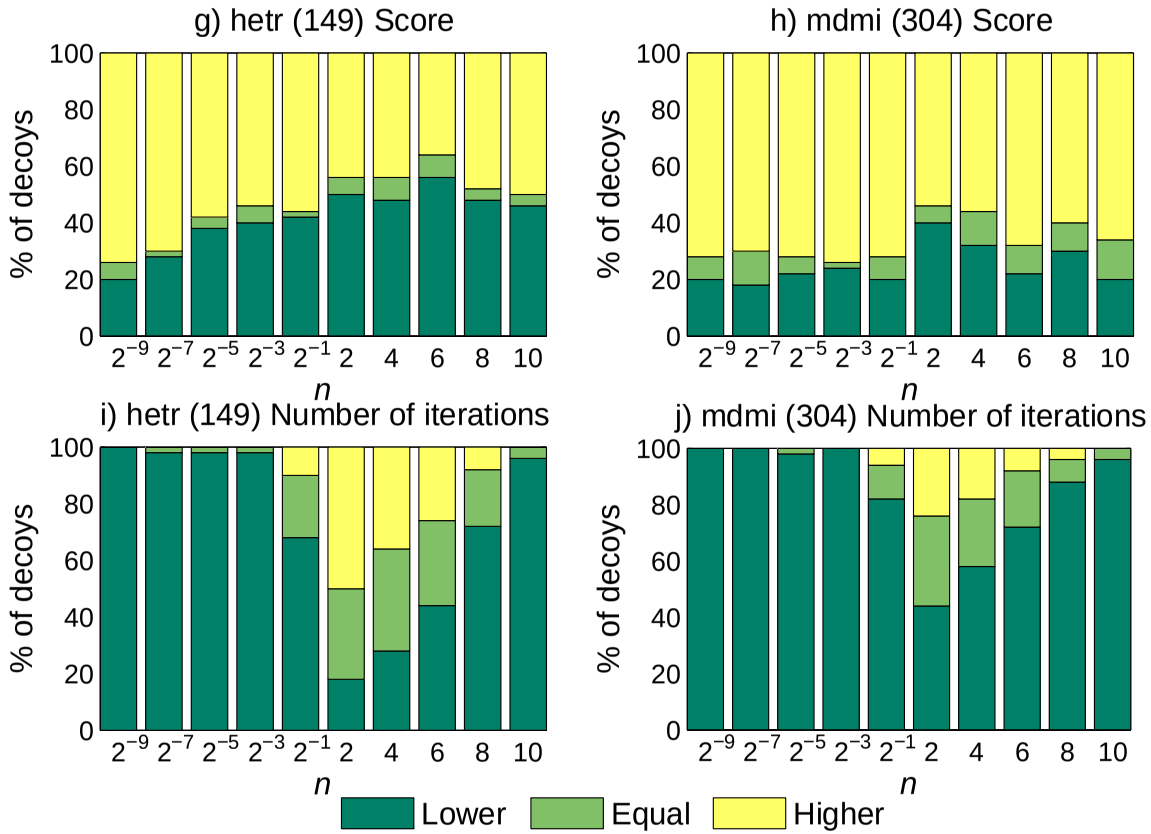

Figure A.2: QNA-XEL: The percentage of decoys with less, equal, higher scores or number of iterations compared to $n = 1$ for proteins hetr (g and i) and mdmi (h and j).

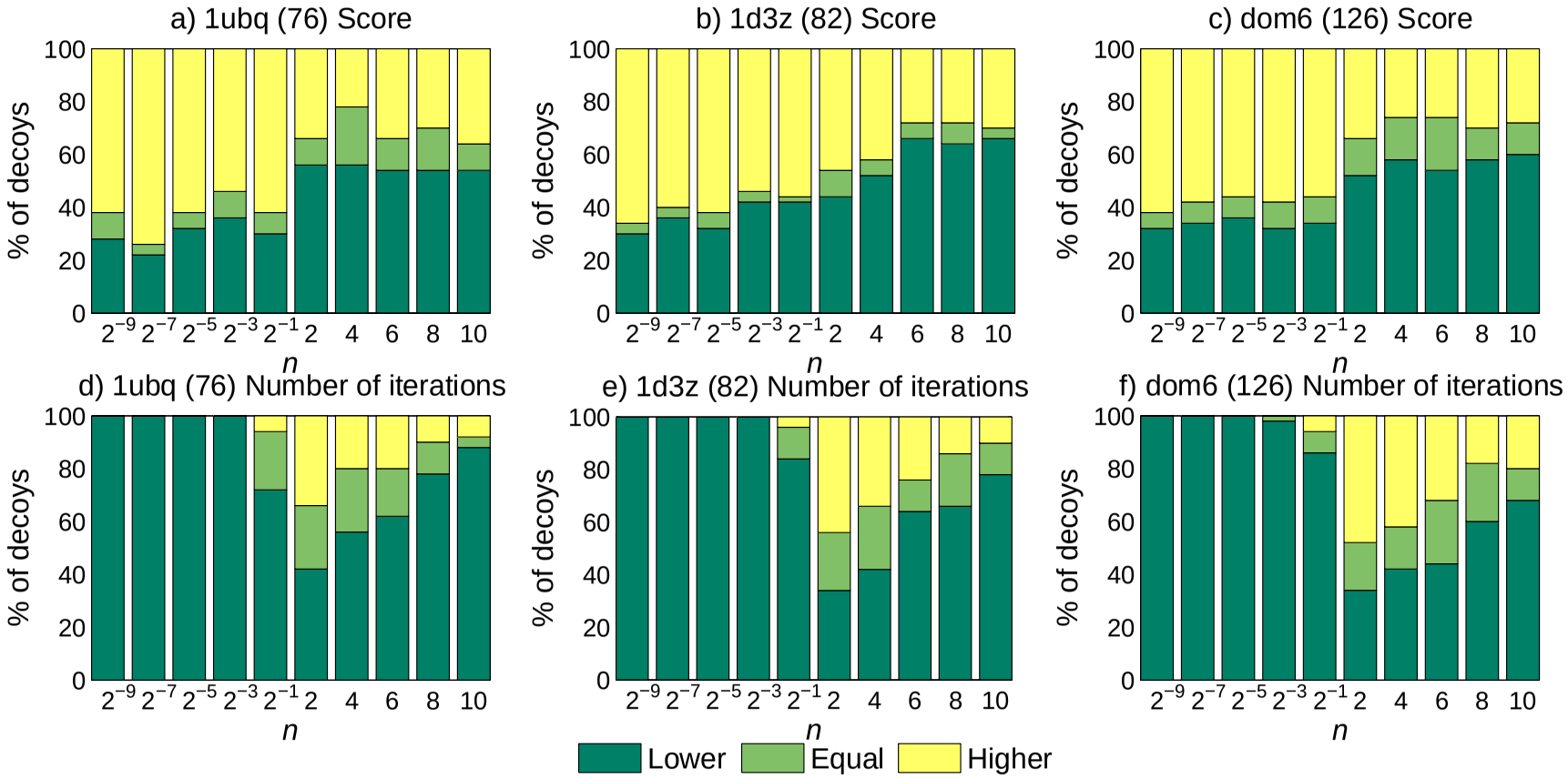

Figure A.3: QNA-XEL (*Hessian only*): The percentage of decoys with less, equal, higher scores or number of iterations compared to $n = 1$ for proteins 1ubq (a and d), 1d3z (b and e), and dom6 (c and f).

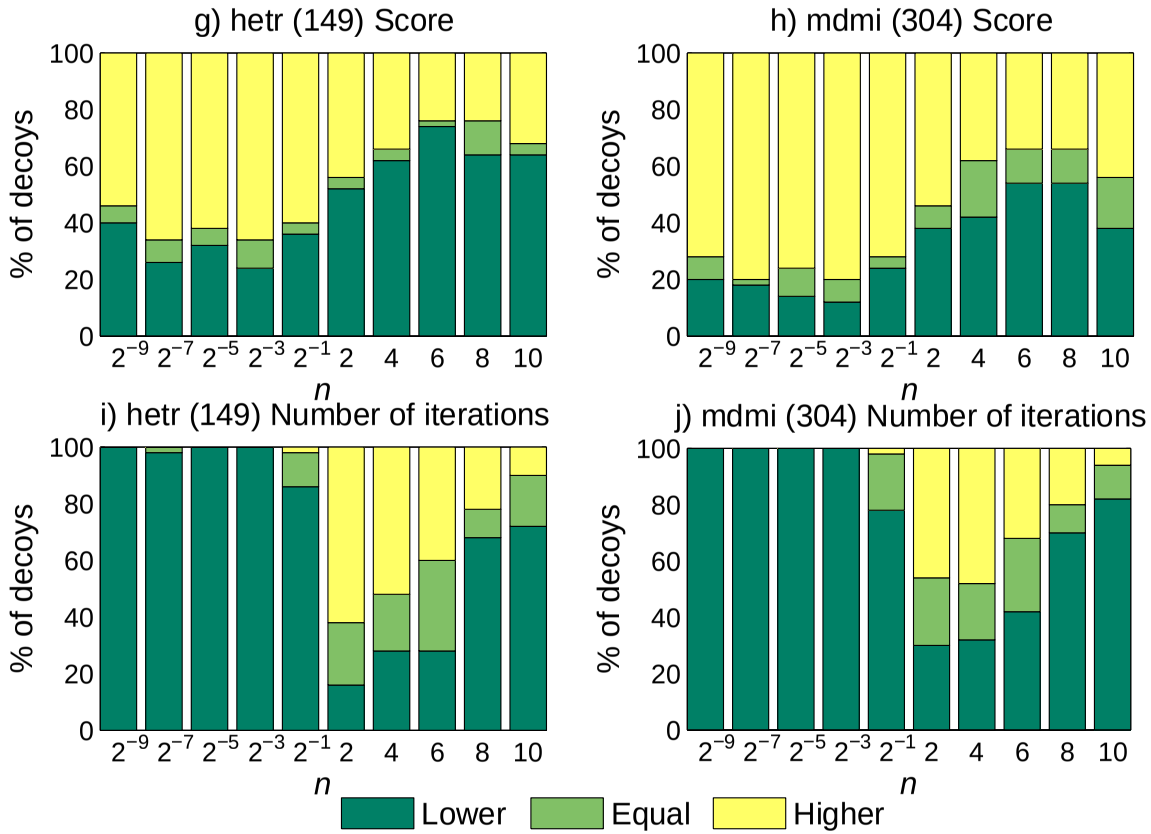

Figure A.4: QNA-XEL (*Hessian only*): The percentage of decoys with less, equal, higher scores or number of iterations compared to $n = 1$ for proteins hetr (g and i) and mdmi (h and j).

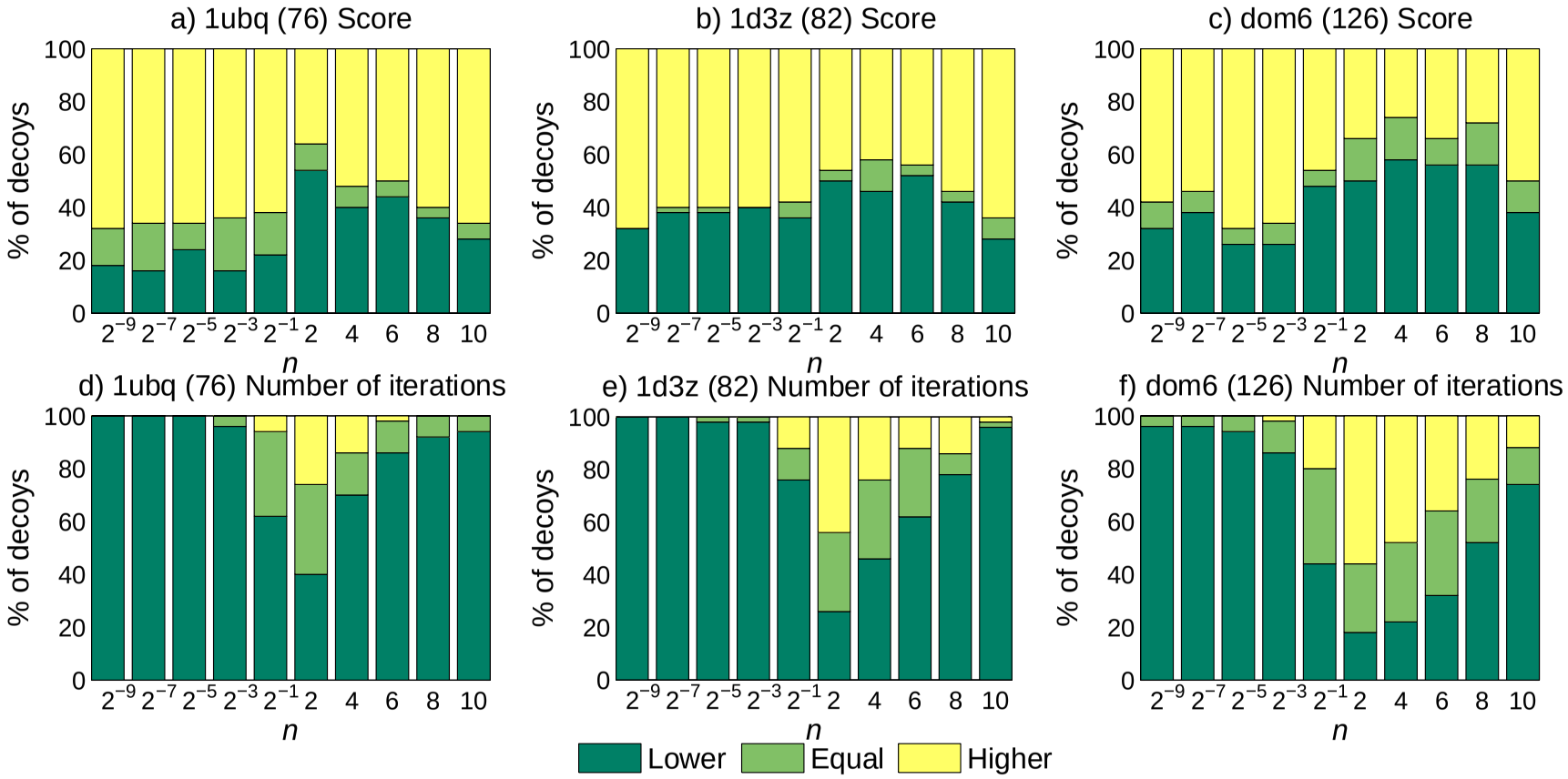

Figure A.5: BFGS-XEL: The percentage of decoys with less, equal, higher scores or number of iterations compared to $n = 1$ for proteins 1ubq (a and d), 1d3z (b and e), and dom6 (c and f).

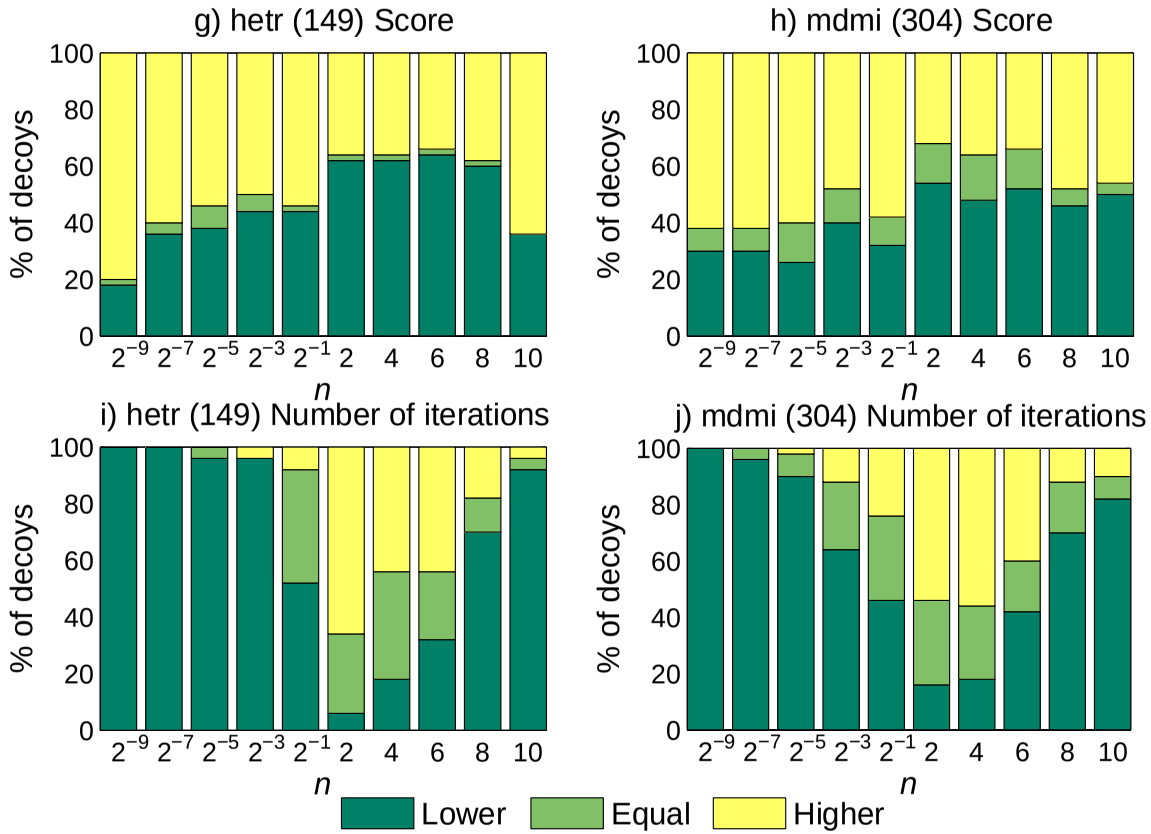

Figure A.6: BFGS-XEL: The percentage of decoys with less, equal, higher scores or number of iterations compared to $n = 1$ for proteins hetr (g and i) and mdmi (h and j).

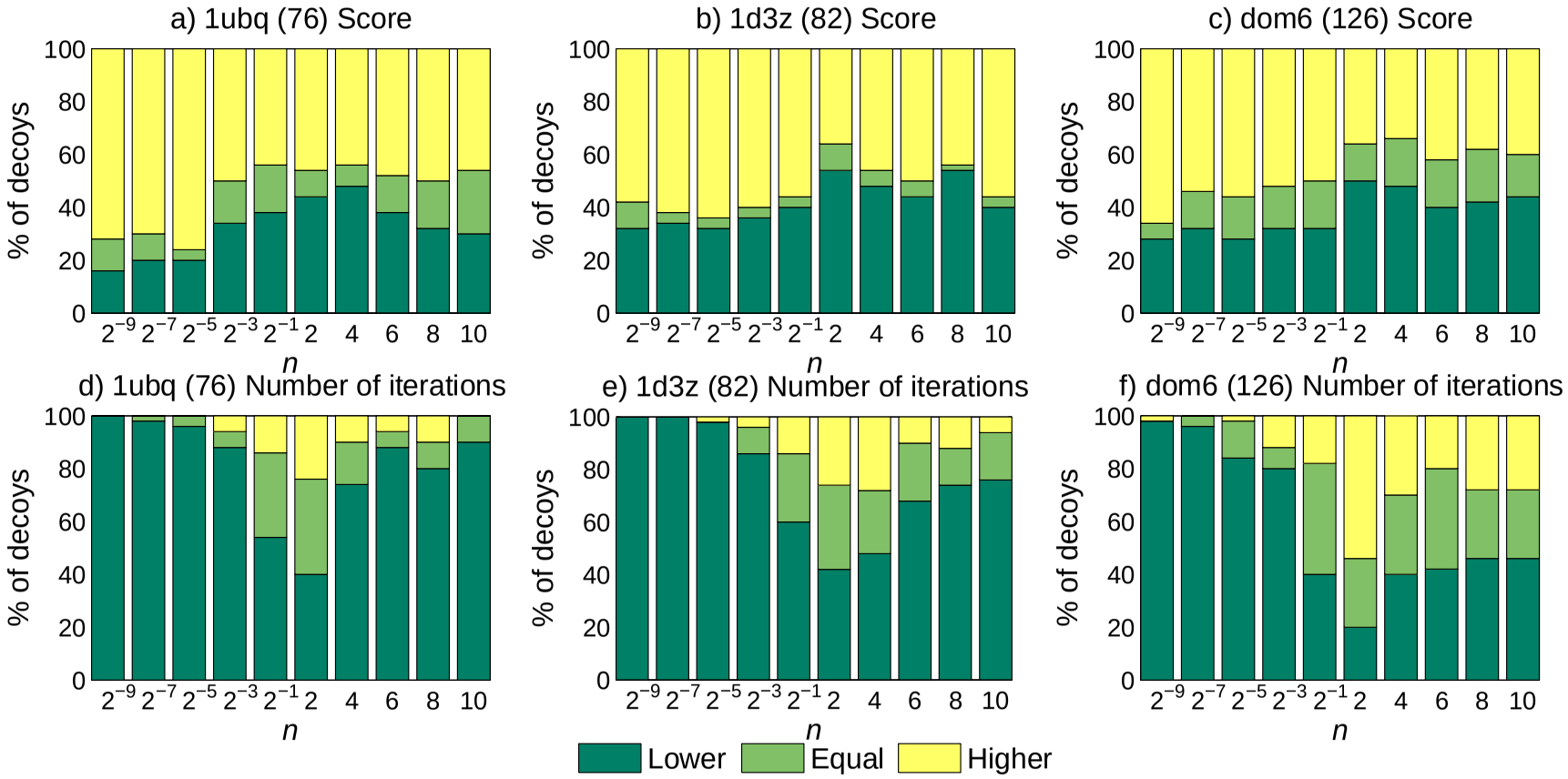

Figure A.7: BFGS-XEL (*Hessian only*): The percentage of decoys with less, equal, higher scores or number of iterations compared to $n = 1$ for proteins 1ubq (a and d), 1d3z (b and e), and dom6 (c and f).

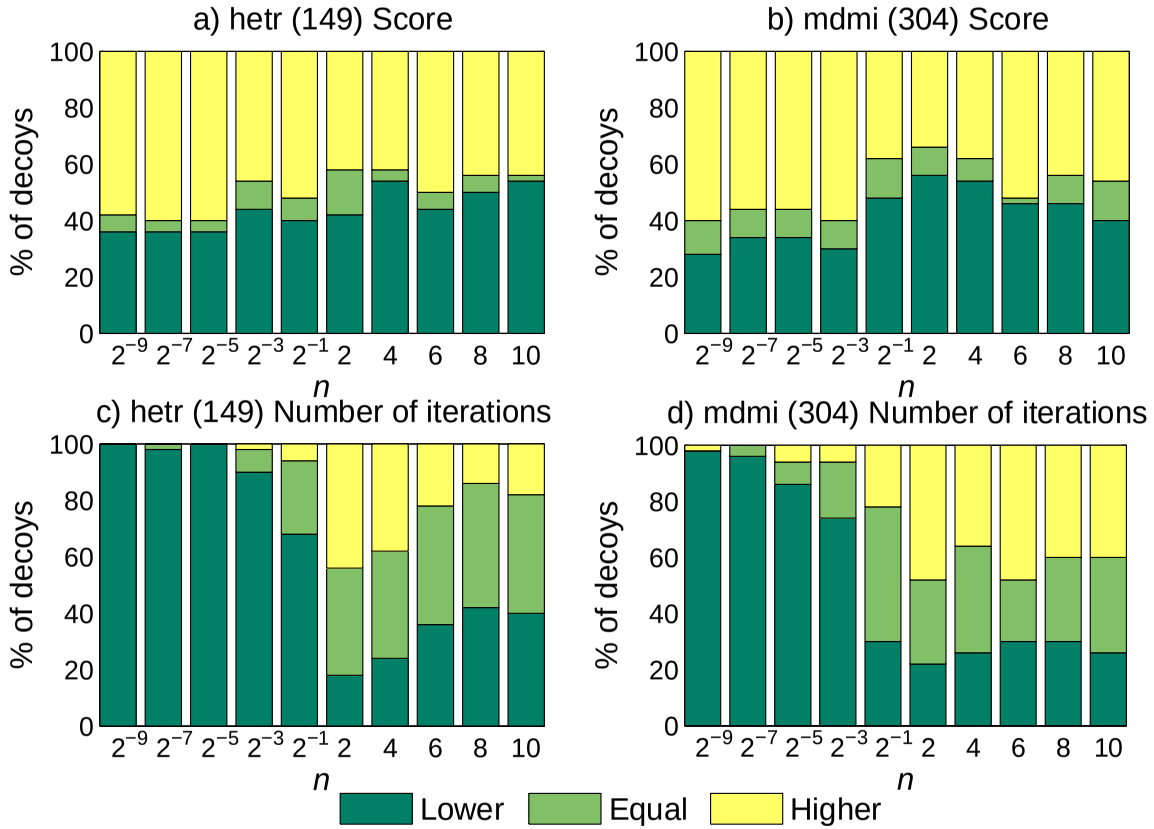

Figure A.8: BFGS-XEL (*Hessian only*): The percentage of decoys with less, equal, higher scores or number of iterations compared to $n = 1$ for proteins hetr (g and i) and mdmi (h and j).

REFERENCES

[1] "Residue count histogram." RCSB Protein Data Bank `http://www.pdb.org`, November 2009.

[2] Alexandrov, V., Lehnert, U., Echols, N., Milburn, D., Engelman, D., and Gerstein, M., "Normal modes for predicting protein motions: A comprehensive database assessment and associated web tool," *Protein Science*, vol. 14, pp. 633–643, 2005.

[3] Alvarado Díaz, C. A., *Computationally Efficient Kinematic Analysis Methods in Biological Structures*. Doctor of philosophy dissertation, Mechanical Engineering, University of Connecticut, Storrs, CT, 2005.

[4] Azmi, A. M., Byrd, R. H., Eskow, E., Schnabel, R. B., Crivelli, S., Philip, T. M., and Head-Gordon, T., "Predicting protein tertiary structure using a global optimization algorithm with smoothing," in *Optimization in Computational Chemistry and Molecular Biology: Local and Global Approaches* (Floudas, C. A. and Pardalos, P. M., eds.), pp. 1–18, Dordrecht/Boston/London: Kluwer Academic, 2000.

[5] Azmi, A. M., Byrd, R. H., Eskow, E., and Schnabel, R. B., "A new smoothing-based global optimization algorithm for protein conformation problems," in *Global Optimization: Scientific and Engineering Case Studies* (Pintér, J. D., ed.), vol. 85 of *Nonconvex Optimization and Its Applications*, pp. 73–102, New York: Springer, 2006.

[6] Belegundu, A. D. and Chandrupatla, T. R., *Optimization Concepts and Applications in Engineering*. Upper Saddle River, N.J.: Prentice Hall, 1999.

[7] Berendsen, H. J. C., "Molecular dynamics simulations: The limits and beyond," in *2nd International Symposium on Algorithms for Macro-Molecular Modelling* (Deuflhard, P., Hermans, J., Leimkuhler, B., Mark, A. E., Reich, S., and Skeel, R. D., eds.), (Berlin, Germany), pp. 3–36, Springer, 1997.

[8] Bertsch, R. A., Vaidehi, N., Chan, S. I., and Goddard, W. A., "Kinetic steps for α-helix formation," *Proteins: Structure, Function, and Genetics*, vol. 33, no. 3, pp. 343–357, 1998.

[9] Bohnenkamp, P., Kazerounian, K., and Ilies, H. T., "Structural prediction of peptide based nano systems via progressive landscape evolution," in *12th IFToMM World Congress*, (Besançon, France), 2007.

[10] Bowers, P. M., Strauss, C. E. M., and Baker, D., "De novo protein structure determination using sparse NMR data.," *Journal of Biomolecular NMR*, vol. 18, no. 4, pp. 311–318, 2000.

[11] Bradley, P., Misura, K. M. S., and Baker, D., "Toward high-resolution de novo structure prediction for small proteins," *Science*, vol. 309, pp. 1868–1871, 2005.

[12] Brooks, B. R., Bruccoleri, R. E., Olafson, B. D., States, D. J., Swaminathan, S., and Karplus, M., "Charmm - a program for macromolecular energy, minimization, and dynamics calculations," *Journal of Computational Chemistry*, vol. 4, no. 2, pp. 187–217, 1983.

[13] Burke, M. G. and Yaliraki, S. N., "Exploring model energy and geometry surfaces using sum of squares decompositions," *Journal of Chemical Theory and Computation*, vol. 2, no. 3, pp. 575–587, 2006.

[14] Caflisch, A. and Karplus, M., "Molecular dynamics studies of protein and peptide folding and unfolding," in *The Protein Folding Problem and Tertiary Structure Prediction* (Merz, K. M. and Grand, S. M. L., eds.), Boston, MA: Birkhäuser, 1994.

[15] Canutescu, A. A. and Dunbrack, R. L., "Cyclic coordinate descent: A robotics algorithm for protein loop closure," *Protein Science*, vol. 12, pp. 963–972, 2003.

[16] Cornell, W., Cieplak, P., Bayly, C., Gould, I., Merz, K., Ferguson, D., Spellmeyer, D., Fox, T., Caldwell, J., and Kollman, P., "A second generation force field for the simulation of proteins, nucleic acids, and organic molecules," *Journal of the American Chemical Society*, vol. 117, pp. 5179–5197, 1995.

[17] Dennis, J. E. and Schnabel., R. B., *Numerical Methods for Unconstrained Optimization and Nonlinear Equations*. Classics in applied mathematics, Philadelphia: Society for Industrial and Applied Mathematics, 1996.

[18] Deuflhard, P., *Newton Methods for Nonlinear Problems: Affine Invariance and Adaptive Algorithms*. Springer Series in Computational Mathematics, Berlin, Germany: Springer, 2006.

[19] Djurdjevic, D. P. and Biggs, M. J., "Ab initio protein fold prediction using evolutionary algorithms: Influence of design and control parameters on performance," *Journal of Computational Chemistry*, vol. 27, no. 11, pp. 1177–1195, 2006.

[20] Dunfield, L. G., Burgess, A. W., and Scheraga, H. A., "Energy parameters in polypeptides. 8. Empirical potential energy algorithm for the conformational analysis of large molecules," *The Journal of Physical Chemistry*, vol. 82, no. 24, pp. 2609–2616, 1978.

[21] Featherstone, R., *Robot Dynamics Algorithms*. The Kluwer International Series in Engineering and Computer Science; Robotics: Vision, Manipulation and Sensors, Boston/Dordrecht/Lancaster: Kluwer Academic Publishers, 1987.

[22] Featherstone, R., "An empirical study of the joint space inertia matrix," *International Journal Robotics Research*, vol. 23, no. 9, pp. 859–871, 2004.

[23] Featherstone, R., "Efficient factorization of the joint space inertia matrix for branched kinematic trees.," *International Journal Robotics Research*, vol. 24, no. 6, pp. 487–500, 2005.

[24] Featherstone, R. and Orin, D. E., "Robot dynamics: Equations and algorithms," in *IEEE International Conference Robotics & Automation*, (San Francisco, CA), pp. 826–834, 2000.

[25] Gupta, K., "Kinematic solutions of robots with continuous three-roll wrists using the zero reference position method," *IEEE*, 1987.

[26] Gupta, K. and Carlson, G. J., "On certain aspects of the zero reference position method and its applications to an industrial manipulator," *IEEE*, 1986.

[27] Hansmann, U. H. E., "Parallel tempering algorithm for conformational studies of biological molecules," *Chemical Physics Letters*, vol. 281, pp. 140–150, 1997.

[28] Jacob, M., Schindler, T., Balbach, J., and Schmid, F. X., "Diffusion control in an elementary protein folding reaction," *Proceedings of The National Academy of Sciences of the United States of America*, vol. 94, pp. 5622–5627, 1997.

[29] Jacobs, D. J., Kuhn, L. A., and Thorpe, M. F., "Flexible and rigid regions in proteins," in *Rigidity Theory and Applications* (Thorpe, M. F. and Duxbury, P., eds.), New York: Kluwer Academic/Plenum, 1999.

[30] Jacobs, D. J., Rader, A., Kuhn, L. A., and Thorpe, M., "Protein flexibility predictions using graph theory," *Proteins: Structure, Function, and Genetics*, vol. 44, pp. 150–165, 2001.

[31] Jain, A., Vaidehi, N., and Rodriguez, G., "A fast recursive algorithm for molecular dynamics simulation," *Journal of Computational Physics*, vol. 106, pp. 258–268, 1993.

[32] Kabsch, W., "A solution for the best rotation to relate two sets of vectors," *Acta crystallographica. Section A, Crystal physics, diffraction, theoretical and general crystallography*, vol. 32, pp. 922–923, 1976.

[33] Kabsch, W., "A discussion of the solution for the best rotation to relate two sets of vectors," *Acta crystallographica. Section A, Crystal physics, diffraction, theoretical and general crystallography*, vol. 34, pp. 827–828, 1978.

[34] Kazerounian, K., Latif, K., and Alvarado, C., "Protofold (Part II): A successive kineto-static compliance method for protein conformation prediction," in *ASME 2004 Design Engineering Technical Conferences and Computers and Information in Engineering Conference*, (Salt Lake City, Utah, USA), 2004.

[35] Kazerounian, K., "From mechanisms and robotics to protein conformation and drug design," *Journal of Mechanical Design*, vol. 126, pp. 40–45, 2004.

[36] Kazerounian, K., Latif, K., Rodriguez, K., and Alvarado, C., "Nano-kinematics for analysis of protein molecules," *Journal of Mechanical Design*, vol. 127, pp. 699–711, 2005.

[37] Kim, D. E., Chivian, D., and Baker, D., "Protein structure prediction and analysis using the Robetta server," *Nucleic Acids Research*, vol. 32, no. Suppl 2:W526–31, 2004.

[38] Kim, M. K., Chirikjian, G. S., and Jernigan, R. L., "Elastic models of conformational transitions in macromolecules," *Journal of Molecular Graphics and Modelling*, vol. 21, pp. 151–160, 2002.

[39] Kim, M. K., Jernigan, R. L., and Chirikjian, G. S., "Rigid-cluster models of conformational transitions in macromolecular machines and assemblies," *Biophysical Journal*, vol. 89, no. 1, pp. 43–55, 2005.

[40] Kirkpatrick, S., Gelatt, C. D., and Vecchi, M. P., "Optimization by simulated annealing," *Science*, vol. 220, no. 4598, pp. 671–680, 1983.

[41] Koh, S. K., Ananthasuresh, G. K., and Croke, C., "A quadratic programming formulation for the design of reduced protein models in continuous sequence space," *Journal of Mechanical Design*, vol. 127, no. 4, pp. 728–735, 2005.

[42] Koh, S. K., Ananthasuresh, G. K., and Vishveshwara, S., "A deterministic optimization approach to protein sequence design using continuous models," *The International Journal of Robotics Research*, vol. 24, no. 2–3, pp. 109–130, 2005.

[43] Lau, K. F. and Dill, K. A., "A lattice statistical mechanics model of the conformational and sequence spaces of proteins," *Macromolecules*, vol. 22, pp. 3986–3997, 1989.

[44] Leach, A. R., *Molecular Modelling: Principles and Applications*. Prentice Hall, 2 ed., 2001.

[45] Levitt, M., "Protein folding by restrained energy minimization and molecular dynamics," *Journal of Molecular Biology*, vol. 170, pp. 723–764, 1983.

[46] Li, Z. and Scheraga, H. A., "Monte Carlo-minimization approach to the multiple-minima problem in protein folding," *Proceedings of the National Academy of Sciences of the United States of America*, vol. 84, no. 19, pp. 6611–6615, 1987.

[47] Mathiowetz, A. M., Jain, A., Karasawa, N., and Goddard, W. A., "Protein simulations using techniques suitable for very large systems: the cell multipole method for nonbond interactions and the Newton-Euler inverse mass operator method for internal coordinate dynamics," *Proteins: Structure, Function, and Genetics*, vol. 20, pp. 227–247, 1994.

[48] Metropolis, N., Rosenbluth, A. W., Rosenbluth, M. N., Teller, A. H., and Teller, E., "Equation of state calculations by fast computing machines," *The Journal of Chemical Physics*, vol. 21, no. 6, pp. 1087–1092, 1953.

[49] Nesterov, Y., *Introductory Lectures on Convex Optimization: A Basic Course*. Applied optimization, Boston: Kluwer Academic Publishers, 2004.

[50] Phillips, A. T., Rosen, J. B., and Dill, K. A., "Energy landscape projections of molecular potential functions," in *Optimization in Computational Chemistry and Molecular Biology: Local and Global Approaches* (Floudas, C. A. and Pardalos, P. M., eds.), pp. 47–55, Dordrecht/Boston/London: Kluwer Academic, 2000.

[51] Piela, L., Kostrowicki, J., and Scheraga, H. A., "On the multiple-minima problem in the conformational analysis of molecules: deformation of the potential energy hypersurface by the diffusion equation method," *The Journal of Physical Chemistry*, vol. 93, no. 8, pp. 3339–3346, 1989.

[52] Press, W. H., Flannery, B. P., Teukolsky, S. A., and Vetterling, W. T., *Numerical Recipes: The art of scientific computing*. Cambridge, UK; New York, NY: Cambridge University Press, 1986.

[53] Rapaport, D. C., "Molecular dynamics simulation of polymer helix formation using rigid-link methods," *Physical Review E*, vol. 66, no. 1, p. 011906, 2002.

[54] Rapaport, D. C., "Dynamics of polymer chain collapse into compact states," *Physical Review E*, vol. 68, no. 4, p. 041801, 2003.

[55] Rapaport, D. C., *The art of molecular dynamics simulation*. Cambridge, UK; New York, NY: Cambridge University Press, 2 ed., 2004.

[56] Rohl, C. A., Strauss, C. E. M., Misura, K. M. S., and Baker, D., "Protein structure prediction using rosetta," *Methods in Enzymology*, vol. 383, pp. 66–93, 2004.

[57] Samudrala, R. and Levitt, M., "Decoys 'R' Us: A database of incorrect conformations to improve protein structure prediction," *Protein Science*, vol. 9, no. 7, pp. 1399–1401, 2000.

[58] Sapsaman, T. and Lipkin, H., "Improving the efficiency of protein conformation prediction with energy landscape modification," in *ASME 2009 International Design Engineering Technical Conferences and Computers and Information in Engineering Conference*, (San Diego, California, USA), 2009.

[59] Schelstraete, S. and Verschelde, H., "Finding minimum-energy configurations of Lennard-Jones clusters using an effective potential," *The Journal of Physical Chemistry A*, vol. 101, no. 3, pp. 310–315, 1997.

[60] Schlick, T., *Molecular Modeling and Simulation*. Springer, 2002.

[61] Sharma, G., Badescu, M., Dubey, A., Mavroidis, C., Tomassone, S. M., and Yarmush, M. L., "Kinematics and workspace analysis of protein based nano-actuators," *Journal of Mechanical Design*, vol. 127, pp. 718–727, 2005.

[62] Siew, N., Elofsson, A., Rychlewski, L., and Fischer, D., "MaxSub: an automated measure for the assessment of protein structure prediction quality," *Bioinformatics*, vol. 16, no. 9, pp. 776–785, 2000.

[63] Simons, K. T., Kooperberg, C., Huang, E., and Baker, D., "Assembly of protein tertiary structures from fragments with similar local sequences using simulated annealing and Bayesian scoring functions," *Journal of Molecular Biology*, vol. 268, no. 1, pp. 209–225, 1997.

[64] Simons, K. T., Ruczinski, I., Kooperberg, C., Fox, B. A., Bystroff, C., and Baker, D., "Improved recognition of native-like protein structures using a combination of sequence-dependent and sequence-independent features of proteins," *Proteins: Structure, Function, and Genetics*, vol. 34, no. 1, pp. 82–95, 1999.

[65] Skolnick, J., "In quest of an empirical potential for protein structure prediction," *Current Opinion in Structural Biology*, vol. 16, no. 2, pp. 166–171, 2006.

[66] Stephenson, D. S. and Binsch, G., "Automated analysis of high-resolution NMR spectra. I. Principles and computational strategy," *Journal of Magnetic Resonance*, vol. 37, no. 3, pp. 395–407, 1980.

[67] Strang, G., *Linear Algebra and Its Applications*. Brooks/Cole, third ed., 1988.

[68] Subramanian, R. and Kazerounian, K., "Residue level inverse kinematics of peptide chains in the presence of observation inaccuracies and bond length changes," in *ASME 2004 Design Engineering Technical Conferences and Computers and Information in Engineering Conference*, (Long Beach, California, USA), 2005.

[69] Subramanian, R. and Kazerounian, K., "Improved molecular model of a peptide unit for proteins," in *ASME 2006 International Design Engineering Technical Conferences and Computers and Information in Engineering Conference*, (Philadelphia, Pennsylvania, USA), 2006.

[70] Swendsen, R. H. and Wang, J.-S., "Replica Monte Carlo simulation of spin-glasses," *Physical Review Letters*, vol. 57, no. 21, pp. 2607–2609, 1986.

[71] Thorpe, M., Lei, M., Rader, A., Jacobs, D. J., and Kuhn, L. A., "Protein flexibility and dynamics using constraint theory," *Journal of Molecular Graphics and Modelling*, vol. 19, pp. 60–69, 2001.

[72] Vaidehi, N., Jain, A., and Goddard, W. A., "Constant temperature constrained molecular dynamics: The Newton-Euler inverse mass operator method," *J. Phys. Chem.*, vol. 100, no. 25, pp. 10508–10517, 1996.

[73] Wales, D. J. and Scheraga, H. A., "Global optimization of clusters, crystals, and biomolecules," *Science*, vol. 285, pp. 1368–1372, 1999.

[74] Wawak, R. J., Wimmer, M. M., and Scheraga, H. A., "Application of the diffusion equation method of global optimization to water clusters," *The Journal of Physical Chemistry*, vol. 96, no. 12, pp. 5138–5145, 1992.

[75] Wells, S., Menor, S., Hespenheide, B., and Thorpe, M. F., "Constrained geometric simulation of diffusive motion in proteins," *Physical Biology*, vol. 2, pp. S127–S136, 2005.

[76] Wroblewska, L. and Skolnick, J., "Can a physics-based, all-atom potential find a protein's native structure among misfolded structures? I. Large scale AMBER benchmarking," *Journal of Computational Chemistry*, vol. 28, no. 12, pp. 2059–2066, 2007.

[77] Yang, L., Tan, C., Hsieh, M. J., Wang, J., Duan, Y., Cieplak, P., Caldwell, J., Kollman, P. A., and Luo, R., "New-generation amber united-atom force field," *The Journal of Physical Chemistry B*, vol. 110, no. 26, pp. 13166–13176, 2006.

[78] Yoon, H.-s., *Optimization Approaches to Protein Folding*. Ph.d. dissertation, Industrial and Systems Engineering, Georgia Institute of Technology, Atlanta, GA, 2006.

[79] Zemla, A., "LGA: a method for finding 3D similarities in protein structures," *Nucleic Acids Research*, vol. 31, no. 13, pp. 3370–3374, 2003.

[80] Zemla, A., Venclovas, C., Moult, J., and Fidelis, K., "Processing and analysis of CASP3 protein structure predictions," *Proteins: Structure, Function, and Genetics*, vol. 37, no. S3, pp. 22–29, 1999.

[81] Zhang, M. and Kavraki, L. E., "A new method for fast and accurate derivation of molecular conformations," *Journal of Chemical Information and Computing Sciences*, vol. 42, pp. 64–70, 2002.

[82] Zhang, M., White, R. A., Wang, L., Goldman, R., Kavraki, L., and Hassett, B., "Improving conformational searches by geometric screening," *Bioinformatics*, vol. 21, no. 5, pp. 624–630, 2005.

[83] Zhang, Y. and Skolnick, J., "Automated structure prediction of weakly homologous proteins on a genomic scale," *Proceedings of The National Academy of Sciences of the United States of America*, vol. 101, no. 20, pp. 7594–7599, 2004.

[84] Zhang, Y. and Skolnick, J., "Tertiary structure predictions on a comprehensive benchmark of medium to large size proteins," *Biophysical Journal*, vol. 87, no. 4, pp. 2647–2655, 2004.

[85] Zhang, Y. and Skolnick, J., "The protein structure prediction problem could be solved using the current PDB library," *Proceedings of The National Academy of Sciences of the United States of America*, vol. 102, no. 4, pp. 1029–1034, 2005.

[86] Zhang, Y. and Skolnick, J., "TM-score: Assessment of quality of protein structure templates and full-length model predictions." Skolnick Research Group: Online Resources `http://cssb.biology.gatech.edu/skolnick/webservice/TM-score/index.shtml`, November 2009.

[87] Zhang, Y., Kihara, D., and Skolnick, J., "Local energy landscape flattening: Parallel hyperbolic Monte Carlo sampling of protein folding," *Proteins: Structure, Function, and Genetics*, vol. 48, pp. 192–201, 2002.

[88] Zhang, Y. and Skolnick, J., "Scoring function for automated assessment of protein structure template quality," *Proteins: Structure, Function, and Bioinformatics*, vol. 57, no. 4, pp. 702–710, 2004.

VITA

Temsiri Sapsaman was born in Nakhonsawan, a province in the middle of Thailand. She grew up in a teachers' housing at a small rural elementary school where her parents worked and she went to school. After fourth grade, she moved to a bigger town in Nakhonsawan to live with her grandparents and became very close to her grandmother. She went to a high school in Bangkok, Triam Udom Suksa school, and stayed with relatives. There she learned about an opportunity to study in the United States. and in 1996, she came to the US and after spending one year in a private prep school, Westover, in Middlebury, CT, she pursued a Bachelor's degree in Mechanical Engineering at Carnegie Mellon University, Pittsburgh, PA. After graduating in May 2001, she moved to Atlanta, GA to pursue her graduate degrees at the Georgia Institute of Technology. There she developed an interest in teaching as she has had several opportunities to work as a graduate teaching assistant and to supervise undergraduate students on their undergraduate research. Also at Georgia Tech, she served on a committee of Woodruff School Graduate Women, a social group for female graduate students in the Woodruff School of Mechanical Engineering, for two years.