Peiyuan (Alexander) Liao

Csef 2018

research notebook 11.0-2.20

Attempt to adapt binary map algorithm to include manipulability measure

Date: 11/9/17

Alexander Liao

Idea 3 (final)

Regressonal amalyses on the predictions by the model

for i=xmin:(xmax-xmin)/itpVal:xmax

for j=ymin:(ymax-ymin)/itpVal:ymax

try

intm=M.ikine6s(transl(i,j,z));

bool=M.maniplty(intm);

catch

bool=0;

end

%{

if all(intm.\*1==intm)

bool=1;

if bool==[]

bool=0;

end

else

bool=0;

end

%}

%coordinateData= horzcat(coordinateData,[i;j;z]);

surfPlan=horzcat(surfPlan,bool);

%size(surfPlan)

end

end

d=[];

for i=1:500

if and(sum(w(i,:)==0)>120,sum(performanceMatrix(i,:)==0)>120)

d=[d,1];

else

k=fitlm(w(i,:),performanceMatrix(i,:));

d=[d,k.Rsquared.ordinary];

end

end

for i=1:500

if isnan(d(i));

d(i)=[];

end

end

Interesting papers:

11/28/17

Revising bibliography

11/10/17 – 12/17/17

Started reading *Deep Learning*

Practicing LaTeX

# Obtaining configuration space and singularity maps for parallel manipulators

[E.MachoO.AltuzarraE.AmezuaA.Hernandez](http://www.sciencedirect.com/science/article/pii/S0094114X09001165?via%3Dihub#!)

## Abstract

The aim of this paper is to describe a general methodology to obtain the entire set of positions that a parallel manipulator can reach and the workspace regions where the robot is controllable. The workspace is computed using a hybrid analytical-discrete procedure. Next the singularity maps are traced by carrying out a kinematic analysis of the positions obtained. To perform the latter a systematic method has been introduced to obtain the corresponding Jacobian matrices. The result of the whole process is the computation of singularity-free workspace regions, associated with certain working and assembly modes. After that, strategies to enlarge the accessible space are easier to plan and implement. This methodology is based on disassembling the manipulator into a mobile platform and a set of kinematic chains.

## Keywords

Parallel manipulator

Workspace

Singularity analysis

Position problem

Working modes

# A new approach to orientation workspace analysis of 6-DOF parallel manipulators

[Ilian A.BonevJehaRyu](http://www.sciencedirect.com/science/article/pii/S0094114X0000032X?via%3Dihub#!)

## Abstract

This paper presents a new discretization method for the computation of the *orientation workspace* of 6-DOF parallel manipulators, defined as the set of all attainable orientations of the mobile platform about a fixed point. The method is based on the use of a modified set of Euler angles and a particular representation of the orientation workspace. In addition, the *projected orientation workspace* is introduced for use in 5-axis applications, defined as the set of possible directions of the approach vector of the mobile platform. Alternative ways of computing these two types of workspaces are also discussed.

# Kinematics and workspace modeling of a new hybrid robot used in minimally invasive surgery

[DoinaPislaAndrasSzilaghyiCalinVaidaNicolae](http://www.sciencedirect.com/science/article/pii/S0736584512001196?via%3Dihub#!)

## Abstract

The geometric and kinematic models of a new surgical hybrid robot used for camera and active instruments positioning are presented in this paper. The robot workspace is computed and illustrated following the singularities analysis. The robot structure consists of two modules: the PARAMIS robot, and the new serial positioning module. The serial positioning module is used to obtain a mechanically fixed remote center of motion (RCM), enabling the structure to manipulate also active instruments. The new robot provides the necessary motion control to respect the particularities and restrictions of surgical applications. The detailed workspace analysis demonstrates the importance of the relative positioning between the robot and the patient. A constructive solution of the new serial module, the numerical results and conclusions from the performed simulations are described.

### Highlights

► A new module and its integration in an existing robotic system are presented. ► The geometric and kinematics models of the new hybrid robot are illustrated. ► The robot workspace is computed and illustrated following the singularities analysis. ► Simulation and numerical results of the inverse kinematics model are obtained. ► These results are validated using the MATLAB/Simulink toolbox as simulator software.

## Keywords

Kinematics

Hybrid robot

Minimally invasive surgery

Robotic module

Workspace modeling

# On the workspace, assembly configurations and singularity curves of the *RRRRR*-type planar manipulator

Author links open overlay panel[J.JesúsCervantes-SánchezJ.CesarHernández-Rodrı́guezJ.GabrielRendón-Sánchez](http://www.sciencedirect.com/science/article/pii/S0094114X99000610?via%3Dihub#!)

## Abstract

In this paper, a broadly applicable approach for numerically obtaining the workspace and the singularity curves of a planar *RRRRR*-type manipulator is presented. The workspace generation is formulated as a direct kinematic problem involving only two branches which are mathematically defined and related with the manipulator's assembly configurations. For solving that problem, the analytical solution of two simple quadratic equations is found. A simple existence criterion is also obtained to detect the set of points forming the manipulator's workspace. On the other hand, the singularity curves are composed of sets of singular points. In order to obtain the singular points, the properties of the Jacobian matrix are used. The complete method has been implemented and tested, as illustrated with examples for different geometrical properties of the manipulator. For each of these examples, the corresponding singularity curves are graphically generated to obtain the practical manipulator's workspace. This feature is a very powerful design tool which is indispensable in visualizing and analyzing the kinematic working capability of the manipulator.

# Determination of the Workspace of 6-DOF Parallel Manipulators

[C. Gosselin](http://mechanicaldesign.asmedigitalcollection.asme.org/solr/searchresults.aspx?author=C.+Gosselin&q=C.+Gosselin)

## Abstract

This paper presents an algorithm for the determination of the workspace of parallel manipulators. The method described here, which is based on geometrical properties of the workspace, leads to a simple graphical representation of the regions of the three-dimensional Cartesian space that are attainable by the manipulator with a given orientation of the platform. Moreover, the volume of the workspace can be easily computed by performing an integration in its boundary, which is obtained from the algorithm. Examples are included to illustrate the application of the method to a six-degree-of-freedom fully parallel manipulator.

Interior and Exterior Boundaries to the Workspace of Mechanical Manipulators

Karim Abdel-Malek

Abstract Analytical methods for identifying the boundary to the workspace of serial mechanical manipulators and the boundary to voids in the workspace are presented. The determination of parametric equations of surface patches that envelop the workspace of serial manipulators was presented elsewhere and is extended in this paper to an analytical method for void identification. Because of the ability to identify closed-form surface patches that exist internal and external to the workspace, a mathematical formulation based on the concept of a normal acceleration function is introduced. Admissible motion in the normal direction to a point on a singular surface is delineated and characterized by definiteness properties of a quadratic form. An enclosure bound by surface patches that do not admit normal motion is identified as a void. Several examples are treated using this formulation to illustrate the method. Keywords: manipulator workspace, voids, singular surfaces, workspace boundary, robotics

NUMERICAL ALGORITHMS FOR MAPPING BOUNDARIES OF MANIPULATOR WORKSPACES

Edward J. Haug Chi-Mei Luh Frederick A. Adkins Jia-Yi Wang

ABSTRACT Numerical algorithms for mapping boundaries of manipulator workspaces are developed and illustrated. Analytical criteria for boundaries of workspaces for both manipulators having the same number of input and output coordinates and redundantly controlled manipulators with a larger number of inputs than outputs are well known, but reliable numerical methods for mapping them have not been presented. In this paper, a numerical method is first developed for finding an initial point on the boundary. From this point, a continuation method that accounts for simple and multiple bifurcation of one dimensional solution curves is developed. Second order Taylor expansions are derived for finding tangents to solution curves at simple bifurcation points of continuation equations and for characterizing barriers to control of manipulators. A recently developed method for tangent calculation at multiple bifurcation points is employed. A planar redundantly controlled serial manipulator is analyzed, determining both the exterior boundary of the accessible output set and interior curves that represent local impediments to motion control. Using these methods, more complex planar and spatial Stewart platform manipulators are analyzed in a companion paper.

## Numerical methods for reachable space generation of humanoid robots(Article)

* Guan, Y.,
* Yokoi, K.,
* Xianmin Zhang

In view of the importance of workspace to robotic design, motion planning and control, we study humanoid reachable spaces. Due to the large number of degrees of freedom, the complexity and special characteristics of humanoid robots that conventional robots do not possess, it would be very difficult or impractical to use analytical or geometric methods to analyze and obtain humanoid reachable spaces. In this paper, we develop two numerical approaches - the optimization-based method and the Monte Carlo method - to generate the reachable space of a humanoid robot. We first formulate the basic constraints (including kinematic constraint and balance constraint) that a humanoid robot must satisfy in manipulation tasks. We then use optimization techniques to build mathematical models for boundary points by which the reachable boundary is formed. This method gives rise to an approximation of the reachable space with accurate boundary points. On the other hand, the Monte Carlo method is relatively simple and more suitable for the visualization of robotic workspace. To utilize the numerical results by the Monte Carlo method, we propose an approach to build a database. We present the algorithms with these two methods and provide illustrating examples conducted on the humanoid HRP-2. © 2008 SAGE Publications Los Angeles.

## Determining manipulator workspace boundaries using the Monte Carlo method and least squares segmentation(Conference Paper)

* Alciatore, David G.,
* Ng, Chung-Ching D.
* View Correspondence (jump link)
* Colorado State Univ, Fort Collins, United States

### Abstract

Many investigators have developed methods for determining the workspace of a manipulator. One method presented by J. Rastegar and D. Perel uses the Monte Carlo method to generate a manipulator's workspace and its approximate boundary surfaces. Since it involves no inverse Jacobian calculation, problems dealing with singularity positions do not exist. However, only a graphical representation of the workspace is provided. The purpose of this work was to further develop the approach in order to determine an analytical description of a general 2D manipulator's workspace. The end result is a series of straight line and arc segments describing the workspace boundaries which were determined and segmented by least-squares line and are fitting methods. Simple end-effector trajectory sweeping is also presented to aid in the segmentation.

# An improved Monte Carlo method based on Gaussian growth to calculate the workspace of robots

Author links open overlay panel[AdriánPeidróÓscarReinosoArturoGilJosé MaríaMarínLuisPayá](http://www.sciencedirect.com/science/article/pii/S0952197617301288#!)

## Abstract

This paper presents a new Monte Carlo method to calculate the workspace of robot manipulators, which we called the Gaussian Growth method. In contrast to classical brute-force Monte Carlo methods, which rely on increasing the number of randomly generated points in the whole workspace to attain higher accuracy, the Gaussian Growth method focuses on populating and improving the precision of poorly defined regions of the workspace. For this purpose, the proposed method first generates an inaccurate seed workspace using a classical Monte Carlo method, and then it uses the Gaussian distribution to densify and grow this seed workspace until the boundaries of the workspace are attained. The proposed method is compared with previous Monte Carlo methods using a 10-degrees-of-freedom robot as a case study, and it is demonstrated that the Gaussian Growth method can generate more accurate workspaces than previous methods requiring the same or less computation time.

## Keywords

Gaussian distribution

Monte Carlo method

Robot manipulator

Workspace

# An Algorithm for the Workspace of a General n-R Robot

[Y. C. Tsai](http://mechanicaldesign.asmedigitalcollection.asme.org/solr/searchresults.aspx?author=Y.+C.+Tsai&q=Y.+C.+Tsai) and [A. H. Soni](http://mechanicaldesign.asmedigitalcollection.asme.org/solr/searchresults.aspx?author=A.+H.+Soni&q=A.+H.+Soni)

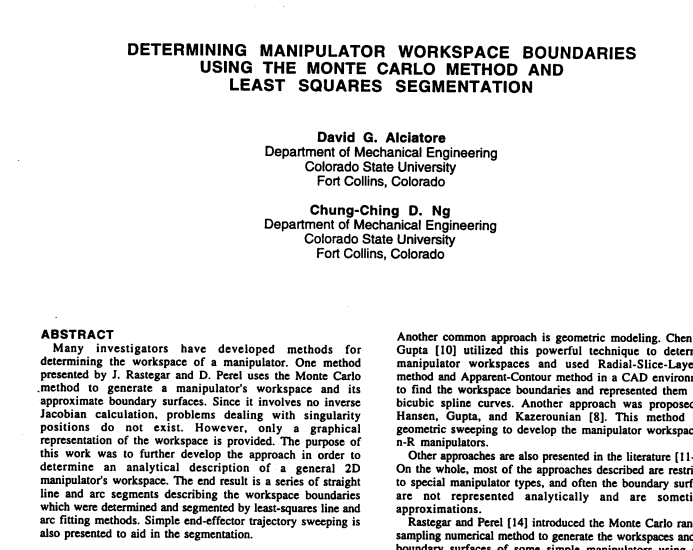
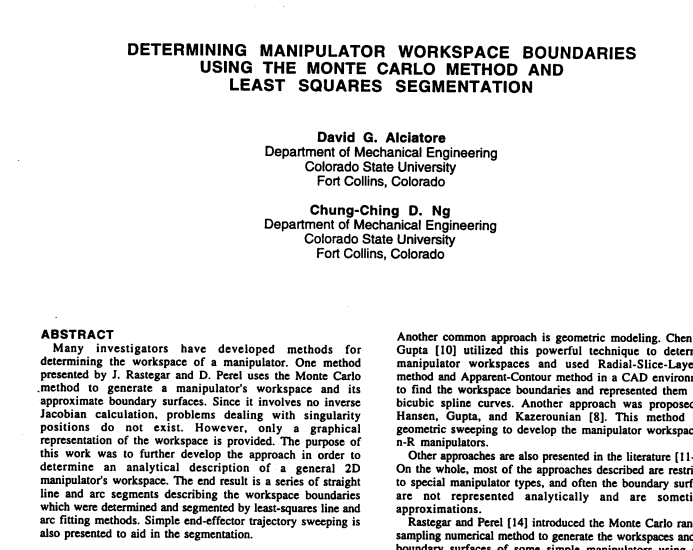
## Abstract

An algorithm is developed to determine the workspace on an arbitrary plane for an n -R robot. The algorithm is based on a linear optimization technique and on small incremental displacements applied to coordinate transformation equations relating the kinematic parameters of the n -R robot. The algorithm provides flexibility to let the user treat the robot hand as a point, a line, or a rigid-body. The revolute pairs of the robot may execute full or partial rotation. The proposed method may be extended to incorporate in a robot the existence of prismatic pairs along with revolute pairs.

# Generation and Evaluation of the Workspace of a Manipulator

[J.A. Hansen](http://journals.sagepub.com/author/Hansen%2C+JA), [K.C. Gupta](http://journals.sagepub.com/author/Gupta%2C+KC), [S.M.K. Kazerounian](http://journals.sagepub.com/author/Kazerounian%2C+SMK)

*An algorithm is presentedfor noniterative generation of the workspace of a general n-R manipulator. It is shown that if special techniques are used for the reduction of points, such algorithms, which otherwise require large storage, become computationally feasible. For the evaluation of the workspace, approach angles and approach lengths for points in the workspace are analyzed by using a stable iterative algorithm for inverse kinematic analysis. One modification makes it a strict descent algorithm with respect to certain measures of deviation from the target position; another uses a variable convergence criterion to improve its speed and accuracy in the determination of the limiting configurations.*



Ceccarelli, Marco & Ottaviano, Erika. (2002). A workspace evaluation of an eclipse robot. Robotica. 20. 299-313. 10.1017/S0263574701003812.

In this paper we have proposed a numerical procedure for determining and evaluating the workspace of the eclipse robot architecture. The eclipse robot is a novel parallel architecture, which has been conceived and designed at the National Seoul University, Korea. The Eclipse robot design has been characterized in term of workspace characteristics, and optimum design solutions have been investigated as functions of the effect of design parameters on workspace.

# On the Evaluation of Manipulator Workspace

[T. W. Lee](http://mechanicaldesign.asmedigitalcollection.asme.org/solr/searchresults.aspx?author=T.+W.+Lee&q=T.+W.+Lee) and [D. C. H. Yang](http://mechanicaldesign.asmedigitalcollection.asme.org/solr/searchresults.aspx?author=D.+C.+H.+Yang&q=D.+C.+H.+Yang)

## Abstract

This paper presents a theorem regarding manipulator workspace and, based on this theorem, a manipulator performance index is introduced. It is found that for a given manipulator structure the ratio of the volume of the workspace to the cube of its total link length is a constant. Algorithms for outlining the boundary profile of workspace and for quantitative evaluation of its volume are presented. A computer package, KAM, is developed, which implements the theories and algorithms developed in this investigation as well as in a companion paper [1]. Several specific examples are given to illustrate the application of the performance index and the capability of KAM.

# A formulation for the workspace boundary of general N-revolute manipulators

[MarcoCeccarelli](http://www.sciencedirect.com/science/article/pii/0094114X9500096H#!)

## Abstract

An algebraic formulation is proposed to describe the workspace boundary of general N-R revolute open chain manipulators. The geometry of the generation process by revolving a figure about an axis has been used to deduce a recursive algorithm for the boundary of ring and hyper-rings by using the envelope concept. The workspace boundary is obtained from the envelope of a torus family which is traced by the parallel circles cut in the boundary of a revolving hyper-ring. The formulation is a function of the dimensional parameters in the manipulator chain and specifically of the last revolute joint angle, only. Some illustrative examples up to a 6R manipulator have been used to test the numerical procedure and they also provide remarks and show some peculiarities of the hyper-ring topology.

# A Fairly General Algorithm to Evaluate Workspace Characteristics of Serial and Parallel Manipulators

[Gianni Castelli](http://www.tandfonline.com/author/Castelli%2C+Gianni),[Erika Ottaviano](http://www.tandfonline.com/author/Ottaviano%2C+Erika) &[**Marco Ceccarelli**](http://www.tandfonline.com/author/Ceccarelli%2C+Marco)

**Abstract**

In this paper a fairly general algorithm is proposed for the determination and evaluation of the workspace both for serial and for parallel manipulators. The procedure is based on a binary representation of the workspace and presents the possibility for a general evaluation of the shape and volume for robot workspace characteristics. Two examples are presented to show the practical feasibility of the proposed procedure and its practical results.

Keywords: [Analysis](http://www.tandfonline.com/keyword/Analysis), [Manipulators](http://www.tandfonline.com/keyword/Manipulators), [Robotics](http://www.tandfonline.com/keyword/Robotics), [Workspace](http://www.tandfonline.com/keyword/Workspace)

# Workspace evaluation of manipulators through finite-partition of *SE*(3)

[YanJinaI-MingChenbGuilinYang](http://www.sciencedirect.com/science/article/pii/S0736584511000202#!)

## Abstract

Workspace analysis and optimization are important in a manipulator design. As the complete workspace of a 6-DOF manipulator is embedded into a 6-D space, it is difficult to quantify and qualify it. Most literatures only considered the 3-D sub workspaces of the complete 6-D workspace. In this paper, a finite-partition approach of the Special Euclidean group *SE*(3) is proposed based on the topology properties of *SE*(3), which is the product of Special Orthogonal group *SO*(3) and *R3*. It is known that the *SO*(3) is homeomorphic to a solid ball *D*3 with antipodal points identified while the geometry of *R3* can be regarded as a cuboid. The complete 6-D workspace *SE*(3) is at the first time parametrically and proportionally partitioned into a number of elements with uniform convergence based on its geometry. As a result, a basis volume element of *SE*(3) is formed by the product of a basis volume element of *R3* and a basis volume element of *SO*(3), which is the product of a basis volume element of *D*3 and its associated integration measure. By this way, the integration of the complete 6-D workspace volume becomes the simple summation of the basis volume elements of *SE*(3). Two new global performance indices, i.e., workspace volume ratio (*Wr*) and global condition index (*GCI*), are defined over the complete 6-D workspace. A newly proposed 3RP PS parallel manipulator is optimized based on this finite-partition approach. As a result, the optimal dimensions for maximal workspace are obtained, and the optimal performance points in the workspace are identified.

## Keywords

Manipulator workspace

Performance measure

Optimization

# Accurate Numerical Methods for Computing 2D and 3D Robot Workspace

Show all authors

[Yi Cao](http://journals.sagepub.com/author/Cao%2C+Yi), [Ke Lu](http://journals.sagepub.com/author/Lu%2C+Ke), [Xiujuan Li](http://journals.sagepub.com/author/Li%2C+Xiujuan)

## Abstract

Exact computation of the shape and size of robot manipulator workspace is very important for its analysis and optimum design. First, the drawbacks of the previous methods based on Monte Carlo are pointed out in the paper, and then improved strategies are presented systematically. In order to obtain more accurate boundary points of two-dimensional (2D) robot workspace, the Beta distribution is adopted to generate random variables of robot joints. And then, the area of workspace is acquired by computing the area of the polygon what is a closed path by connecting the boundary points together. For comparing the errors of workspaces which are generated by the previous and the improved method from shape and size, one planar robot manipulator is taken as example. A spatial robot manipulator is used to illustrate that the methods can be used not only on planar robot manipulator, but also on the spatial. The optimal parameters are proposed in the paper to computer the shape and size of 2D and 3D workspace. Finally, we provided the computation time and discussed the generation of 3D workspace which is based on 3D reconstruction from the boundary points.

**Keywords** [Beta distribution](http://journals.sagepub.com/keyword/Beta+Distribution), [Robot manipulator](http://journals.sagepub.com/keyword/Robot+Manipulator), [Polygon area](http://journals.sagepub.com/keyword/Polygon+Area), [2D and 3D workspace](http://journals.sagepub.com/keyword/2D+And+3D+Workspace), [Shape and size](http://journals.sagepub.com/keyword/Shape+And+Size)

First version of MATLAB scripts to generate samples in huge batches:

Date: 12/18/17

Alexander Liao

Idea 3 (final)

1. Parallel computing suitability
2. Function to generate binary map for a single manipulator and to be called repetitively
3. Script to interpret the result

**5-DOF script for generation**

%{

Author: Alexander Liao Kent'20

Library: MATLAB RTB

Description: Discretization method of serial-link manipulator workspace

calculation by binary maps

5-DOF Spherical wrist with shoulder offset

Constant Orientation Workspace

\\Data Generation for 0Neural Network//

%}

clear all;

n=1; %Number of samples

entry=[ -1 1 -1 1 -1 1 .1 .1 .1 ];

% x-range y-range z range deltaX deltaY deltaZ

xmin=entry(1);

xmax=entry(2);

ymin=entry(3);

ymax=entry(4);

zmin=entry(5);

zmax=entry(6);

deltaX=entry(7);

deltaY=entry(8);

deltaZ=entry(9);

orientation = [ pi/2 pi/2 0];

% Ang

ular displacement x-axis y z (in radians)

angleX=orientation(1);

angleY=orientation(2);

angleZ=orientation(3);

i=xmin;

j=ymin;

k=zmin;

c1=1;

c2=1;

c3=1;

H=cell( int32(((xmax-xmin)/deltaX)+1), int32(((ymax-ymin)/deltaY)+1), int32(((zmax-zmin)/deltaZ)+1) );

%The coordinates of the node is stored in a 3-D cell array

for k=zmin:deltaZ:zmax

c2=1;

for j=ymin:deltaY:ymax

c1=1;

for i=xmin:deltaX:xmax

H{c1, c2, c3}=[i,j,k];

c1=c1+1;

end

c2=c2+1;

end

c3=c3+1;

end

%Assigning values to the cell array

%Initializing

[ M, bMap ] = binaryMap5DOFR3(H,xmax,xmin,ymax,ymin,zmax,zmin,deltaX,deltaY,deltaZ,angleX,angleY,angleZ);

%Input vector corresponding binary map

[~,sM]=size(M);

[~,s]=size(bMap);

%Creating arrays to store the results

resM=zeros(n,sM);

resB=zeros(n,s);

resM(1,:)=M;

resB(1,:)=bMap;

parfor num=2:n

[M,bMap]= binaryMap5DOFR3(H,xmax,xmin,ymax,ymin,zmax,zmin,deltaX,deltaY,deltaZ,angleX,angleY,angleZ);

resM(num,:)=M;

resB(num,:)=bMap;

end

[~,l]=size(bMap);

filename = horzcat('5DOFR3.','n',num2str(n),'.','L',num2str(l),'.',datestr(datetime("now"),'mm.dd.yy.HH.MM'),'.mat');

% Type Number of Samples Date

save(filename,'resM','resB');

%The coordinates of the node is stored in a 3-D cell array

for k=zmin:deltaZ:zmax

c2=1;

for j=ymin:deltaY:ymax

c1=1;

for i=xmin:deltaX:xmax

H{c1, c2, c3}=[i,j,k];

c1=c1+1;

end

c2=c2+1;

end

c3=c3+1;

end

%Assigning values to the cell array

%Initializing

[ M, bMap ] = binaryMap5DOFR3(H,xmax,xmin,ymax,ymin,zmax,zmin,deltaX,deltaY,deltaZ,angleX,angleY,angleZ);

%Input vector corresponding binary map

[~,sM]=size(M);

[~,s]=size(bMap);

%Creating arrays to store the results

resM=zeros(n,sM);

resB=zeros(n,s);

resM(1,:)=M;

resB(1,:)=bMap;

parfor num=2:n

[M,bMap]= binaryMap5DOFR3(H,xmax,xmin,ymax,ymin,zmax,zmin,deltaX,deltaY,deltaZ,angleX,angleY,angleZ);

resM(num,:)=M;

resB(num,:)=bMap;

end

[~,l]=size(bMap);

filename = horzcat('5DOFR3.','n',num2str(n),'.','L',num2str(l),'.',datestr(datetime("now"),'mm.dd.yy.HH.MM'),'.mat');

% Type Number of Samples Date

save(filename,'resM','resB');

function [M,bMap]= binaryMap5DOFR3(H,xmax,xmin,ymax,ymin,zmax,zmin,deltaX,deltaY,deltaZ,angleX,angleY,angleZ)

%Randomized DH standard parameter for manipulator

%{

D= [0 0 rand()/2 0 rand()/2 ];

A= [0 rand()/2 rand()/20 0 0 ];

alpha=[pi/2 0 0 -pi/2 pi/2];

%}

l1=Link([0 D(1) A(1) alpha(1)]);

l2=Link([0 D(2) A(2) alpha(2)]);

l3=Link([0 D(3) A(3) alpha(3)]);

l4=Link([0 D(4) A(4) alpha(4)]);

l5=Link([0 D(5) A(5) alpha(5)]);

m=SerialLink([l1,l2,l3,l4,l5]);

P=zeros(size(H));

%Binary matrix

[sizeX,sizeY,sizeZ]=size(P);

%Optional PUMA560 Model for demonstration purposes

%mdl\_puma560;

%m=p560;

%Determining if the point is within the workspace of robot

for k=1:sizeZ

for j=1:sizeY

for i=1:sizeX

try

if all(not(isempty(m.ikine( trotx(angleX) \* troty(angleY) \* trotz(angleZ)\* transl(H{i,j,k}), 'ilimit',250,'tol',5e-9,'mask',[1 1 1 0 0 0] ))))

% Numerical Inverse Kinematics Homogeneous Transformation Matrix

P(i,j,k)=1;

end

catch

end

end

end

end

M=[D,A,alpha];

%DH parameters

cdData=[xmin:deltaX:xmax,ymin:deltaY:ymax,zmin:deltaZ:zmax];

M=[M,cdData];

bMap=zeros(1,sizeX\*sizeY\*sizeZ);

%Foiling the binary map into a vector

index=1;

for k=1:sizeZ

for j=1:sizeY

for i=1:sizeX

bMap(1,index)=P(i,j,k);

index=index+1;

end

end

end

end

**Interpreting the results from the data**

%Please load the ".mat" file first

num=1;

M=resM(num,:);

B=resB(num,:);

[~,s]=size(M);

D=M(1:6);

A=M(7:12);

alpha=M(13:18);

Coordinates=M(19:s);

stopPt=[];

dif=M(20)-M(19);

for i=20:s-1

if round((M(i+1)-M(i)),2)~=round(dif,2)

stopPt=horzcat(stopPt,i);

dif=M(i+2)-M(i+1);

end

end

if size(stopPt)==[1 1]

stopPt=horzcat(stopPt,18+(2\*(s-18)/3)+1);

elseif isempty(stopPt)

stopPt=[18+((s-18)/3)+1, 18+(2\*(s-18)/3)+1];

end

xArray=M(19:stopPt(1));

yArray=M(stopPt(1)+1:stopPt(2));

zArray=M(stopPt(2)+1:s);

[~,sX]=size(xArray);

[~,sY]=size(yArray);

[~,sZ]=size(zArray);

xmin=xArray(1);

xmax=xArray(sX);

ymin=yArray(1);

ymax=yArray(sY);

zmin=zArray(1);

zmax=zArray(sZ);

deltaX=xArray(2)-xArray(1);

deltaY=yArray(2)-yArray(1);

deltaZ=zArray(2)-zArray(1);

i=xmin;

j=ymin;

k=zmin;

c1=1;

c2=1;

c3=1;

index=1;

for k=zmin:deltaZ:zmax

c2=1;

for j=ymin:deltaY:ymax

c1=1;

for i=xmin:deltaX:xmax

H{c1, c2, c3}=[i,j,k];

c1=c1+1;

end

c2=c2+1;

end

c3=c3+1;

end

[sizeX,sizeY,sizeZ]=size(H);

P=zeros(size(H));

for k=1:sizeZ

for j=1:sizeY

for i=1:sizeX

P(i,j,k)=B(index);

index=index+1;

end

end

end

delP=zeros(size(P));

for k=2:sizeZ-1

for j=2:sizeY-1

for i=2:sizeX-1

if and(sum(sum(sum(P(i-1:i+1, j-1:j+1, k-1:k+1))))<27, P(i,j,k)==1)

delP(i,j,k)=1;

end

%{

if and(sum(sum(sum(P(i-1:i+1, j-1:j+1, k-1:k+1))))==27, P(i,j,k)==1)

delP(i,j,k)=0;

end

if P(i,j,k)==0

delP(i,j,k)=0;

end

%}

end

end

end

l1=Link([0 D(1) A(1) alpha(1)]);

l2=Link([0 D(2) A(2) alpha(2)]);

l3=Link([0 D(3) A(3) alpha(3)]);

l4=Link([0 D(4) A(4) alpha(4)]);

l5=Link([0 D(5) A(5) alpha(5)]);

l6=Link([0 D(6) A(6) alpha(6)]);

m=SerialLink([l1,l2,l3,l4,l5,l6]);

xData=zeros(1,sizeX\*sizeY\*sizeZ);

yData=zeros(size(xData));

zData=zeros(size(xData));

binMap=zeros(size(xData));

binBound=zeros(size(xData));

dexMap=zeros(size(xData));

ct=1;

for k=1:sizeZ

for j=1:sizeY

for i=1:sizeX

xData(ct)=H{i,j,k}(1);

yData(ct)=H{i,j,k}(2);

zData(ct)=H{i,j,k}(3);

binMap(ct)=P(i,j,k);

binBound(ct)=delP(i,j,k);

ct=ct+1;

end

end

end

fXData=zeros(size(xData));

fYData=zeros(size(xData));

fZData=zeros(size(xData));

[~,lBin]=size(binMap);

for i=1:lBin

if binMap(i)>0

fXData(i)=xData(i);

fYData(i)=yData(i);

fZData(i)=zData(i);

end

end

ct2=1;

[~,lF]=size(fXData);

while ct2<=lF

if all([fXData(ct2)==0;fYData(ct2)==0;fZData(ct2)==0])

fXData(ct2)=[];

fYData(ct2)=[];

fZData(ct2)=[];

dexMap(ct2)=[];

binMap(ct2)=[];

binBound(ct2)=[];

else

ct2=ct2+1;

end

[~,lF]=size(fXData);

end

for i=1:sizeZ

area(i)=sum(sum(P(:,:,i))\*deltaX)\*deltaY;

end

volume=sum(area)\*deltaZ;

figure

m.plot([pi/4 pi/4 pi/4 pi/4 pi/4 pi/4]);

hold on;

scatter3(fXData,fYData,fZData,20,binBound,'filled')

figure

scatter3(fXData,fYData,fZData,20,binMap,'filled')

%Foiling the binary map into a vector

index=1;

for k=1:sizeZ

for j=1:sizeY

for i=1:sizeX

bMap(1,index)=P(i,j,k);

index=index+1;

end

end

end

end

i=xmin;

j=ymin;

k=zmin;

c1=1;

c2=1;

c3=1;

index=1;

for k=zmin:deltaZ:zmax

c2=1;

for j=ymin:deltaY:ymax

c1=1;

for i=xmin:deltaX:xmax

H{c1, c2, c3}=[i,j,k];

c1=c1+1;

end

c2=c2+1;

end

c3=c3+1;

end

[sizeX,sizeY,sizeZ]=size(H);

P=zeros(size(H));

for k=1:sizeZ

for j=1:sizeY

for i=1:sizeX

P(i,j,k)=B(index);

index=index+1;

end

end

end

delP=zeros(size(P));

for k=2:sizeZ-1

for j=2:sizeY-1

for i=2:sizeX-1

if and(sum(sum(sum(P(i-1:i+1, j-1:j+1, k-1:k+1))))<27, P(i,j,k)==1)

delP(i,j,k)=1;

end

%{

if and(sum(sum(sum(P(i-1:i+1, j-1:j+1, k-1:k+1))))==27, P(i,j,k)==1)

delP(i,j,k)=0;

end

if P(i,j,k)==0

delP(i,j,k)=0;

end

%}

end

end

end

l1=Link([0 D(1) A(1) alpha(1)]);

l2=Link([0 D(2) A(2) alpha(2)]);

l3=Link([0 D(3) A(3) alpha(3)]);

l4=Link([0 D(4) A(4) alpha(4)]);

l5=Link([0 D(5) A(5) alpha(5)]);

l6=Link([0 D(6) A(6) alpha(6)]);

m=SerialLink([l1,l2,l3,l4,l5,l6]);

xData=zeros(1,sizeX\*sizeY\*sizeZ);

yData=zeros(size(xData));

zData=zeros(size(xData));

binMap=zeros(size(xData));

binBound=zeros(size(xData));

dexMap=zeros(size(xData));

ct=1;

for k=1:sizeZ

for j=1:sizeY

for i=1:sizeX

xData(ct)=H{i,j,k}(1);

yData(ct)=H{i,j,k}(2);

zData(ct)=H{i,j,k}(3);

binMap(ct)=P(i,j,k);

binBound(ct)=delP(i,j,k);

ct=ct+1;

end

end

end

fXData=zeros(size(xData));

fYData=zeros(size(xData));

fZData=zeros(size(xData));

[~,lBin]=size(binMap);

for i=1:lBin

if binMap(i)>0

fXData(i)=xData(i);

fYData(i)=yData(i);

fZData(i)=zData(i);

end

end

ct2=1;

[~,lF]=size(fXData);

while ct2<=lF

if all([fXData(ct2)==0;fYData(ct2)==0;fZData(ct2)==0])

fXData(ct2)=[];

fYData(ct2)=[];

fZData(ct2)=[];

dexMap(ct2)=[];

binMap(ct2)=[];

binBound(ct2)=[];

else

ct2=ct2+1;

end

[~,lF]=size(fXData);

end

for i=1:sizeZ

area(i)=sum(sum(P(:,:,i))\*deltaX)\*deltaY;

end

volume=sum(area)\*deltaZ;

figure

m.plot([pi/4 pi/4 pi/4 pi/4 pi/4 pi/4]);

hold on;

scatter3(fXData,fYData,fZData,20,binBound,'filled')

figure

scatter3(fXData,fYData,fZData,20,binMap,'filled')

l1=Link([0 D(1) A(1) alpha(1)]);

l2=Link([0 D(2) A(2) alpha(2)]);

l3=Link([0 D(3) A(3) alpha(3)]);

l4=Link([0 D(4) A(4) alpha(4)]);

l5=Link([0 D(5) A(5) alpha(5)]);

l6=Link([0 D(6) A(6) alpha(6)]);

m=SerialLink([l1,l2,l3,l4,l5,l6]);

xData=zeros(1,sizeX\*sizeY\*sizeZ);

yData=zeros(size(xData));

zData=zeros(size(xData));

binMap=zeros(size(xData));

binBound=zeros(size(xData));

dexMap=zeros(size(xData));

ct=1;

for k=1:sizeZ

for j=1:sizeY

for i=1:sizeX

xData(ct)=H{i,j,k}(1);

yData(ct)=H{i,j,k}(2);

zData(ct)=H{i,j,k}(3);

binMap(ct)=P(i,j,k);

binBound(ct)=delP(i,j,k);

ct=ct+1;

end

end

end

fXData=zeros(size(xData));

fYData=zeros(size(xData));

fZData=zeros(size(xData));

[~,lBin]=size(binMap);

for i=1:lBin

if binMap(i)>0

fXData(i)=xData(i);

fYData(i)=yData(i);

fZData(i)=zData(i);

end

end

ct2=1;

[~,lF]=size(fXData);

while ct2<=lF

if all([fXData(ct2)==0;fYData(ct2)==0;fZData(ct2)==0])

fXData(ct2)=[];

fYData(ct2)=[];

fZData(ct2)=[];

dexMap(ct2)=[];

binMap(ct2)=[];

binBound(ct2)=[];

else

ct2=ct2+1;

end

[~,lF]=size(fXData);

end

for i=1:sizeZ

area(i)=sum(sum(P(:,:,i))\*deltaX)\*deltaY;

end

volume=sum(area)\*deltaZ;

figure

m.plot([pi/4 pi/4 pi/4 pi/4 pi/4 pi/4]);

hold on;

scatter3(fXData,fYData,fZData,20,binBound,'filled')

figure

scatter3(fXData,fYData,fZData,20,binMap,'filled')

while ct2<=lF

if all([fXData(ct2)==0;fYData(ct2)==0;fZData(ct2)==0])

fXData(ct2)=[];

fYData(ct2)=[];

fZData(ct2)=[];

dexMap(ct2)=[];

binMap(ct2)=[];

binBound(ct2)=[];

else

ct2=ct2+1;

end

[~,lF]=size(fXData);

end

for i=1:sizeZ

area(i)=sum(sum(P(:,:,i))\*deltaX)\*deltaY;

end

volume=sum(area)\*deltaZ;

figure

m.plot([pi/4 pi/4 pi/4 pi/4 pi/4 pi/4]);

hold on;

scatter3(fXData,fYData,fZData,20,binBound,'filled')

figure

scatter3(fXData,fYData,fZData,20,binMap,'filled')

Attempt to generate manipulability map data

%Initializing

[ M, bMap ] = binaryMap6DOFR3Mani (H,xmax,xmin,ymax,ymin,zmax,zmin,deltaX,deltaY,deltaZ,angleX,angleY,angleZ);

Date: 12/19/17

Alexander Liao

Idea 3 (final)

function [M,bMap]= binaryMap6DOFR3Mani(H,xmax,xmin,ymax,ymin,zmax,zmin,deltaX,deltaY,deltaZ,angleX,angleY,angleZ)

%Randomized DH standard parameter for manipulator

D=[0 0 rand()/2 rand()/2 0 0];

A=[0 rand()/2 rand()/20 0 0 0];

alpha=[ pi/2 0 -pi/2 pi/2 -pi/2 0];

l1=Link([0 D(1) A(1) alpha(1)]);

l2=Link([0 D(2) A(2) alpha(2)]);

l3=Link([0 D(3) A(3) alpha(3)]);

l4=Link([0 D(4) A(4) alpha(4)]);

l5=Link([0 D(5) A(5) alpha(5)]);

l6=Link([0 D(6) A(6) alpha(6)]);

P=zeros(size(H));

%Binary matrix

[sizeX,sizeY,sizeZ]=size(P);

m=SerialLink([l1,l2,l3,l4,l5,l6]);

%Optional PUMA560 Model for demonstration purposes

mdl\_puma560;

m=p560;

%Determining if the point is within the workspace of robot

for k=1:sizeZ

for j=1:sizeY

for i=1:sizeX

try

if all(not(isnan(m.ikine6s( trotx(angleX) \* troty(angleY) \* trotz(angleZ)\* transl(H{i,j,k}) ))))

% Analytical Inverse Kinematics Homogeneous Transformation Matrix

P(i,j,k)=m.maniplty(m.ikine6s( trotx(angleX) \* troty(angleY) \* trotz(angleZ)\* transl(H{i,j,k})));

end

catch

end

end

end

end

M=[D,A,alpha];

%DH parameters

cdData=[xmin:deltaX:xmax,ymin:deltaY:ymax,zmin:deltaZ:zmax];

M=[M,cdData];

bMap=zeros(1,sizeX\*sizeY\*sizeZ);

%Foiling the binary map into a vector

index=1;

for k=1:sizeZ

for j=1:sizeY

for i=1:sizeX

bMap(1,index)=P(i,j,k);

index=index+1;

end

end

end

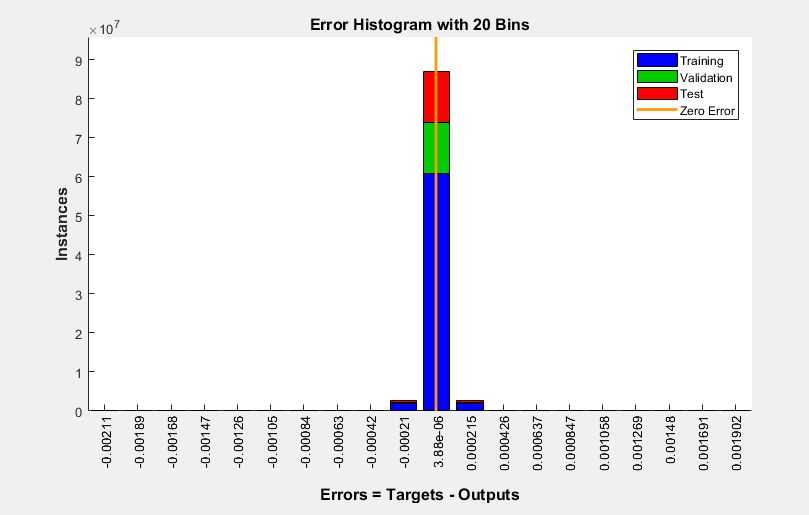
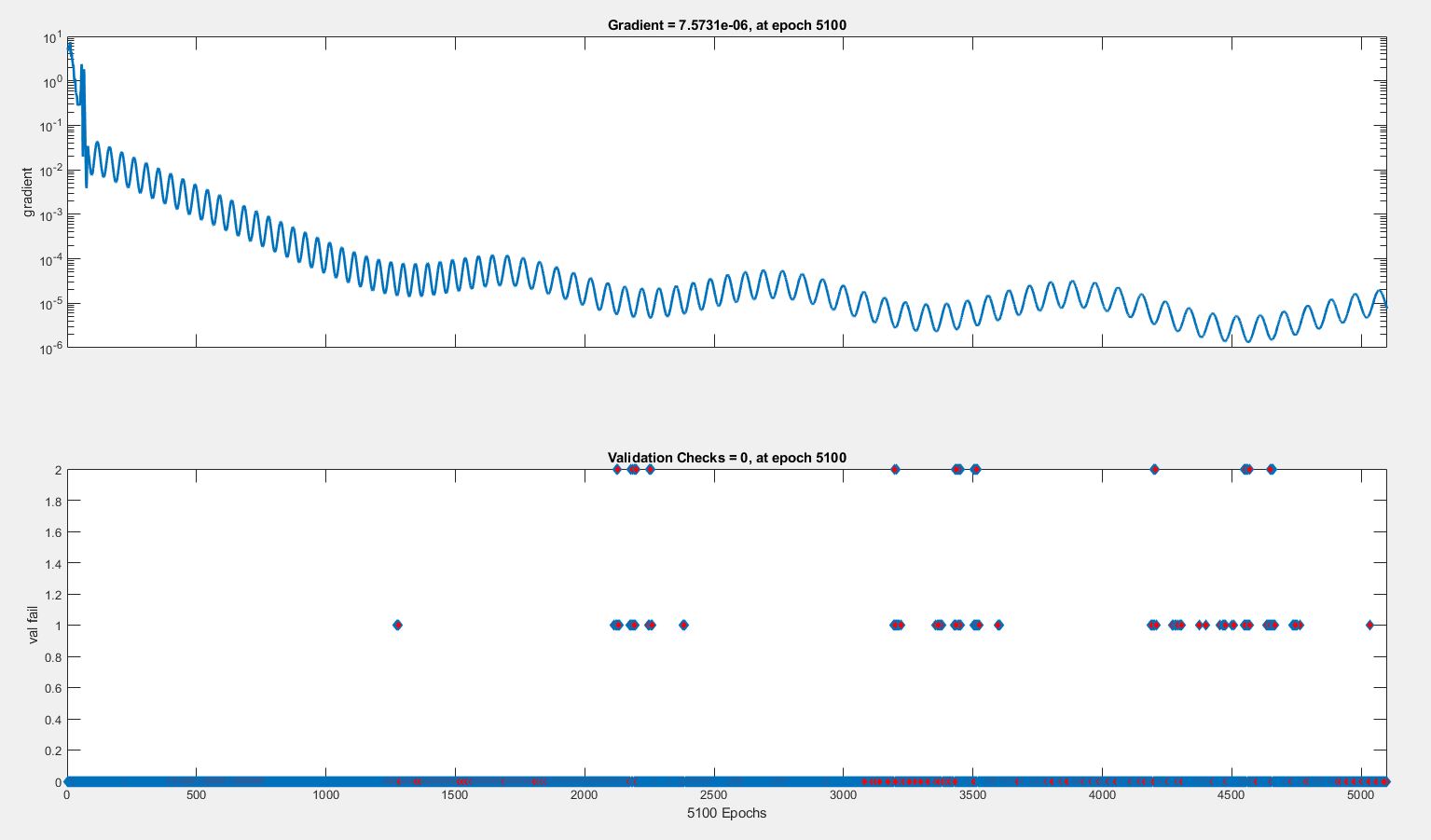
end

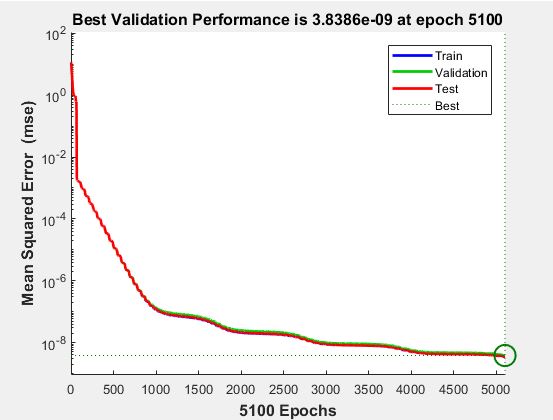
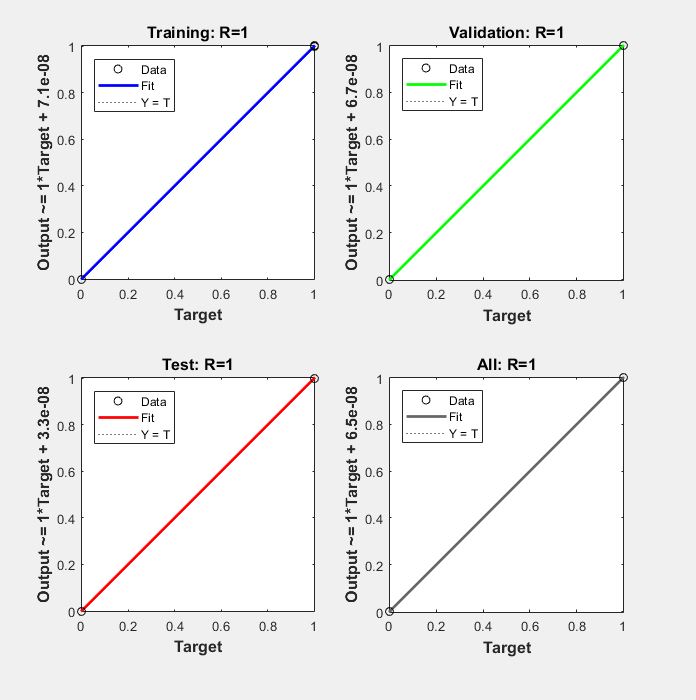
Started model training with scaled conjugate gradient

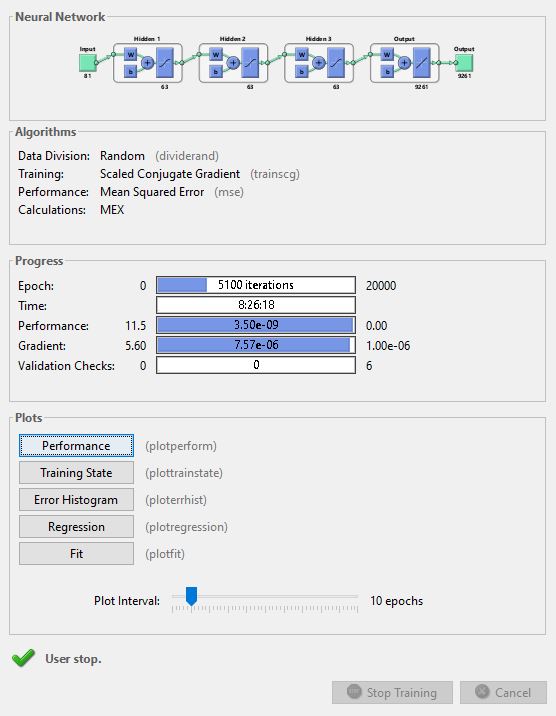
Date: 12/20/17

Alexander Liao

Idea 3 (final)





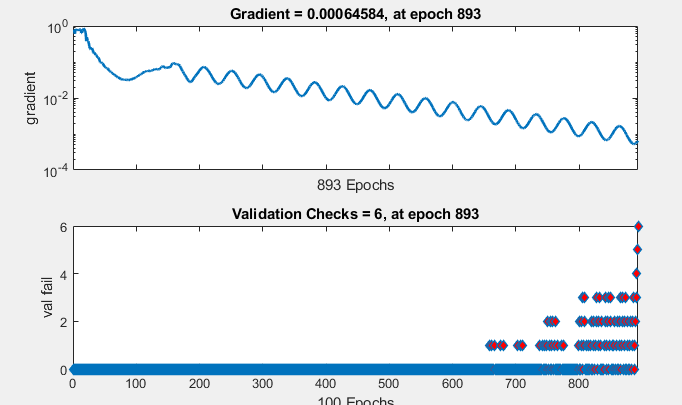
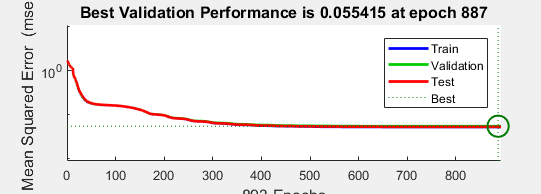


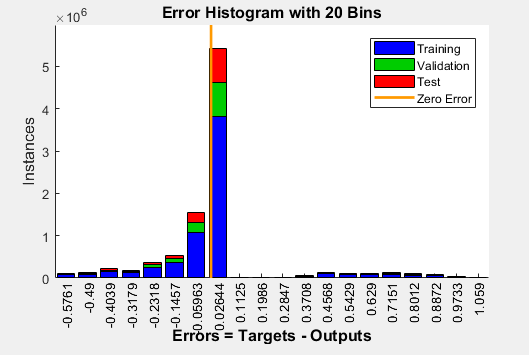
Scaled conjugate gradient training session #2; started generating training data on parallel computing platform

Date: 12/20/17

Alexander Liao

Idea 3 (final)

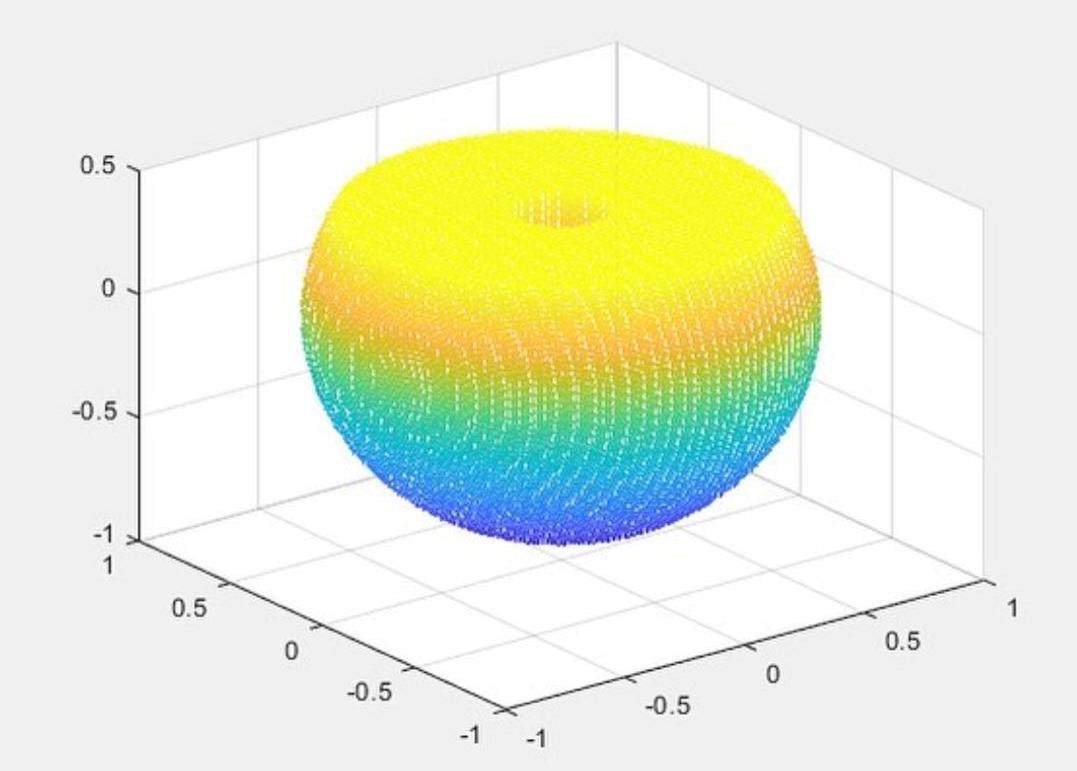


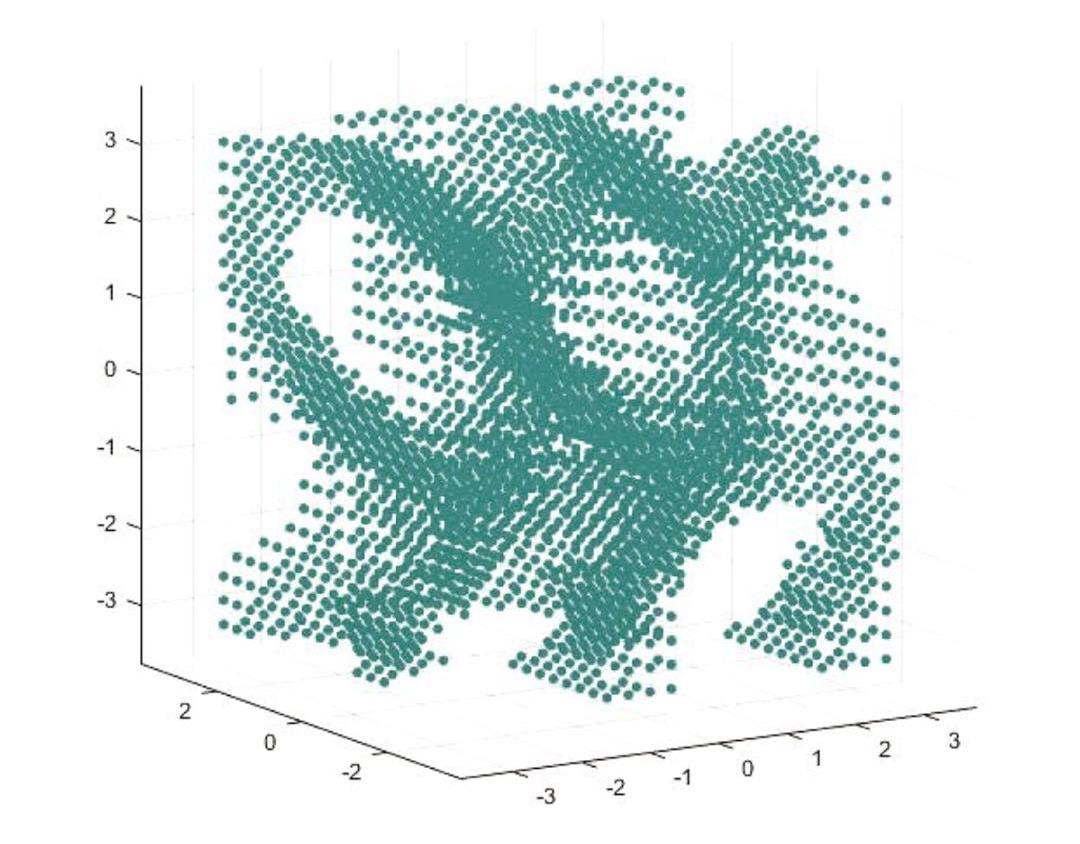


Date: 12/25/17

Alexander Liao

Idea 3 (final)

Started writing paper; finished visualization of constant orientation workspace of PUMA560 with delta =0.01; visualization of orientation workspace of a 6-DOF manipulator



Started first revision for data generation

Date: 1/4/2018

Alexander Liao

Idea 3 (final)

1. 6DOFR3

a. Purpose 1: suitability for multiple precisions (different # of partitions)

[0 0 0]

[ -1 1 -1 1 -1 1 1 1 1 ] 1e4

[ -1 1 -1 1 -1 1 .5 .5 .5 ] 5e4

[ -1 1 -1 1 -1 1 .1 .1 .1 ] 2e5

[ -1 1 -1 1 -1 1 .08 .08 .08 ] 3e5

b. Purpose 2: different scopes

[0 0 0]

[ 0 2 -1 1 -1 1 .1 .1 .1 ] 2e5

[ -1 1 0 2 -1 1 .1 .1 .1 ] 2e5

[ -1 1 -1 1 0 2 .1 .1 .1 ] 2e5

[ -.5 1.5 -.5 1.5 -.5 1.5 .1 .1 .1 ] 2e5

[ 0 1 0 1 0 1 .1 .1 .1 ] 1e5

[ -1.5 1.5 -1.5 1.5 -1.5 1.5 .1 .1 .1 ] 3e5

c. Purpose 3: different pts

[ -1 1 -1 1 -1 1 1 1 1 ] 1e4

[pi/2 0 0]

[0 pi/2 0]

[0 0 pi/2]

[pi/4 pi/4 pi/4]

2. 6DOFSO3

a. Purpose 3: suitability for orientation workspace

[.5 .5 .5]

[ 0 2pi 0 2pi 0 2pi pi/10 pi/10 pi/10 ] 2e5

[ 0 2pi 0 2pi 0 2pi pi/3 pi/3 pi/3 ] 5e4

[ 0 pi 0 pi 0 pi pi/10 pi/10 pi/10 ] 1e5

[1 0 0]

[ 0 2pi 0 2pi 0 2pi pi/8 pi/8 pi/8 ] 1e5

3. 5DOFR3

[0 0 0]

[ -1 1 -1 1 -1 1 .5 .5 .5 ] 2.5e4

[0 0 0]

[ 0 1 0 1 0 1 .5 .5 .5 ] 1e4

[0 0 0]

[ -1 1 -1 1 -1 1 1 1 1 ] 5e3

[pi/4 0 0]

[ -1 1 -1 1 -1 1 .5 .5 .5 ] 2.5e4

4. 5DOFSO3

[.5 .5 .5]

[ 0 2pi 0 2pi 0 2pi pi/3 pi/3 pi/3 ] 2.5e4

[ 0 pi 0 pi 0 pi pi/3 pi/3 pi/3 ] 1e4

[ 0 2pi 0 2pi 0 2pi pi/2 pi/2 pi/2 ]2e4

[1 0 0]

[ 0 2pi 0 2pi 0 2pi pi/3 pi/3 pi/3 ] 2.5e4

\*repeat 2

5. Purpose 5: suitability for mixed up scopes

[ (random x/y/z/ min/max with max-min =2 ) .1 .1 .1 ] 30000

[ (random x/y/z/ min/max with max-min =1 ) .1 .1 .1 ] 20000

[ (random x/y/z/ min/max with max-min =.5 ) .1 .1 .1 ] 10000

Purpose 6: suitability for mixed up precisions

[all random with number of partitions being constant in the end] 30000

Final version of data generation plan

Date: 1/8/2018

Alexander Liao

Idea 3 (final)

DataGeneration6DOFR3(5e4,[-1 1 -1 1 -1 1 1 1 1] ,[0 0 0])

DataGeneration6DOFR3(5e4,[-1 1 -1 1 -1 1 .5 .5 .5] ,[0 0 0])

DataGeneration6DOFR3(5e4,[-1 1 -1 1 -1 1 .1 .1 .1] ,[0 0 0])

DataGeneration6DOFR3(5e4,[-1 1 -1 1 -1 1 .1 .1 .1] ,[pi/2 0 0])

DataGeneration6DOFR3(5e4,[-1 1 -1 1 -1 1 .1 .1 .1] ,[0 pi/2 0])

DataGeneration6DOFR3(5e4,[-1 1 -1 1 -1 1 .1 .1 .1] ,[0 0 pi/2])

DataGeneration6DOFR3(5e4,[-1 1 -1 1 -1 1 .1 .1 .1] ,[pi/4 pi/4 pi/4])

DataGeneration6DOFR3(5e4,[-1 1 -1 1 -1 1 .08 .08 .08] ,[0 0 0])

DataGeneration6DOFR3(5e4,[0 2 -1 1 -1 1 .1 .1 .1] ,[0 0 0])

DataGeneration6DOFR3(5e4,[-1 1 -1 1 -1 1 .1 .1 .1] ,[0 0 0])

DataGeneration6DOFR3(5e4,[-1 1 0 2 -1 1 .1 .1 .1] ,[0 0 0])

DataGeneration6DOFR3(5e4,[-1 1 -1 1 0 2 .1 .1 .1] ,[0 0 0])

DataGeneration6DOFR3(5e4,[-.5 1.5 -.5 1.5 -.5 1.5 .1 .1 .1] ,[0 0 0])

DataGeneration6DOFR3(5e4,[-1.5 1.5 -1.5 1.5 -1.5 1.5 .1 .1 .1] ,[0 0 0])

DataGeneration6DOFSO3(5e4,[ 0 2\*pi 0 2\*pi 0 2\*pi pi/10 pi/10 pi/10 ],[.5 .5 .5])

DataGeneration6DOFSO3(5e4, [ 0 2\*pi 0 2\*pi 0 2\*pi pi/3 pi/3 pi/3 ],[.5 .5 .5])

DataGeneration6DOFSO3(5e4, [ 0 pi 0 pi 0 pi pi/10 pi/10 pi/10 ],[.5 .5 .5])

DataGeneration6DOFSO3(5e4,[ 0 2\*pi 0 2\*pi 0 2\*pi pi/8 pi/8 pi/8 ],[1 0 0])

DataGeneration5DOFR3(5e4,[-1 1 -1 1 -1 1 .5 .5 .5] ,[0 0 0])

DataGeneration5DOFR3(5e4,[0 1 0 1 0 1 .5 .5 .5] ,[0 0 0])

DataGeneration5DOFR3(5e4,[-1 1 -1 1 -1 1 1 1 1] ,[0 0 0])

DataGeneration5DOFR3(5e4,[-1 1 -1 1 -1 1 .5 .5 .5] ,[pi/4 0 0])

DataGeneration5DOFSO3(5e4,[ 0 2\*pi 0 2\*pi 0 2\*pi pi/3 pi/3 pi/3] ,[.5 .5 .5])

DataGeneration5DOFSO3(5e4,[ 0 pi 0 pi 0 pi pi/3 pi/3 pi/3] ,[.5 .5 .5])

DataGeneration5DOFSO3(5e4,[ 0 2\*pi 0 2\*pi 0 2\*pi pi/2 pi/2 pi/2] ,[.5 .5 .5])

DataGeneration5DOFSO3(5e4,[0 2\*pi 0 2\*pi 0 2\*pi pi/3 pi/3 pi/3] ,[1 0 0])

DataGeneration6DOFR3(5e4,[-1 1 -1 1 -1 1 1 1 1] ,[0 0 0])

DataGeneration6DOFR3(5e4,[-1 1 -1 1 -1 1 .5 .5 .5] ,[0 0 0])

DataGeneration6DOFR3(5e4,[-1 1 -1 1 -1 1 .1 .1 .1] ,[0 0 0])

DataGeneration6DOFR3(5e4,[-1 1 -1 1 -1 1 .1 .1 .1] ,[pi/2 0 0])

DataGeneration6DOFR3(5e4,[-1 1 -1 1 -1 1 .1 .1 .1] ,[0 pi/2 0])

DataGeneration6DOFR3(5e4,[-1 1 -1 1 -1 1 .1 .1 .1] ,[0 0 pi/2])

DataGeneration6DOFR3(5e4,[-1 1 -1 1 -1 1 .1 .1 .1] ,[pi/4 pi/4 pi/4])

DataGeneration6DOFR3(5e4,[-1 1 -1 1 -1 1 .08 .08 .08] ,[0 0 0])

DataGeneration6DOFR3(5e4,[0 2 -1 1 -1 1 .1 .1 .1] ,[0 0 0])

DataGeneration6DOFR3(5e4,[-1 1 -1 1 -1 1 .1 .1 .1] ,[0 0 0])

DataGeneration6DOFR3(5e4,[-1 1 0 2 -1 1 .1 .1 .1] ,[0 0 0])

DataGeneration6DOFR3(5e4,[-1 1 -1 1 0 2 .1 .1 .1] ,[0 0 0])

DataGeneration6DOFR3(5e4,[-.5 1.5 -.5 1.5 -.5 1.5 .1 .1 .1] ,[0 0 0])

DataGeneration6DOFR3(5e4,[-1.5 1.5 -1.5 1.5 -1.5 1.5 .1 .1 .1] ,[0 0 0])

DataGeneration6DOFSO3(5e4,[ 0 2\*pi 0 2\*pi 0 2\*pi pi/10 pi/10 pi/10 ],[.5 .5 .5])

DataGeneration6DOFSO3(5e4, [ 0 2\*pi 0 2\*pi 0 2\*pi pi/3 pi/3 pi/3 ],[.5 .5 .5])

DataGeneration6DOFSO3(5e4, [ 0 pi 0 pi 0 pi pi/10 pi/10 pi/10 ],[.5 .5 .5])

DataGeneration6DOFSO3(5e4,[ 0 2\*pi 0 2\*pi 0 2\*pi pi/10 pi/10 pi/10 ],[1 0 0])

DataGeneration5DOFR3(5e4,[-1 1 -1 1 -1 1 .5 .5 .5] ,[0 0 0])

DataGeneration5DOFR3(5e4,[0 1 0 1 0 1 .5 .5 .5] ,[0 0 0])

DataGeneration5DOFR3(5e4,[-1 1 -1 1 -1 1 1 1 1] ,[0 0 0])

DataGeneration5DOFR3(5e4,[-1 1 -1 1 -1 1 .5 .5 .5] ,[pi/4 0 0])

DataGeneration5DOFSO3(5e4,[ 0 2\*pi 0 2\*pi 0 2\*pi pi/3 pi/3 pi/3] ,[.5 .5 .5])

DataGeneration5DOFSO3(5e4,[ 0 pi 0 pi 0 pi pi/3 pi/3 pi/3] ,[.5 .5 .5])

DataGeneration5DOFSO3(5e4,[ 0 2\*pi 0 2\*pi 0 2\*pi pi/2 pi/2 pi/2] ,[.5 .5 .5])

DataGeneration5DOFSO3(5e4,[ 0 2\*pi 0 2\*pi 0 2\*pi pi/3 pi/3 pi/3] ,[1 0 0])

DataGeneration6DOFR3(5e4,[-1 1 -1 1 -1 1 1 1 1] ,[0 0 0])

DataGeneration6DOFR3(5e4,[-1 1 -1 1 -1 1 .5 .5 .5] ,[0 0 0])

DataGeneration6DOFR3(5e4,[-1 1 -1 1 -1 1 .1 .1 .1] ,[0 0 0])

DataGeneration6DOFR3(5e4,[-1 1 -1 1 -1 1 .1 .1 .1] ,[pi/2 0 0])

DataGeneration6DOFR3(5e4,[-1 1 -1 1 -1 1 .1 .1 .1] ,[0 pi/2 0])

DataGeneration6DOFR3(5e4,[-1 1 -1 1 -1 1 .1 .1 .1] ,[0 0 pi/2])

DataGeneration6DOFR3(5e4,[-1 1 -1 1 -1 1 .1 .1 .1] ,[pi/4 pi/4 pi/4])

DataGeneration6DOFR3(5e4,[-1 1 -1 1 -1 1 .08 .08 .08] ,[0 0 0])

DataGeneration6DOFR3(5e4,[0 2 -1 1 -1 1 .1 .1 .1] ,[0 0 0])

DataGeneration6DOFR3(5e4,[-1 1 -1 1 -1 1 .1 .1 .1] ,[0 0 0])

DataGeneration6DOFR3(5e4,[-1 1 0 2 -1 1 .1 .1 .1] ,[0 0 0])

DataGeneration6DOFR3(5e4,[-1 1 -1 1 0 2 .1 .1 .1] ,[0 0 0])

DataGeneration6DOFR3(5e4,[-.5 1.5 -.5 1.5 -.5 1.5 .1 .1 .1] ,[0 0 0])

DataGeneration6DOFR3(5e4,[-1.5 1.5 -1.5 1.5 -1.5 1.5 .1 .1 .1] ,[0 0 0])

DataGeneration6DOFSO3(5e4,[ 0 2\*pi 0 2\*pi 0 2\*pi pi/10 pi/10 pi/10 ],[.5 .5 .5])

DataGeneration6DOFSO3(5e4, [ 0 2\*pi 0 2\*pi 0 2\*pi pi/3 pi/3 pi/3 ],[.5 .5 .5])

DataGeneration6DOFSO3(5e4, [ 0 pi 0 pi 0 pi pi/10 pi/10 pi/10 ],[.5 .5 .5])

DataGeneration6DOFSO3(5e4,[ 0 2\*pi 0 2\*pi 0 2\*pi pi/8 pi/8 pi/8 ],[1 0 0])

DataGeneration5DOFR3(5e4,[-1 1 -1 1 -1 1 .5 .5 .5] ,[0 0 0])

DataGeneration5DOFR3(5e4,[0 1 0 1 0 1 .5 .5 .5] ,[0 0 0])

DataGeneration5DOFR3(5e4,[-1 1 -1 1 -1 1 1 1 1] ,[0 0 0])

DataGeneration5DOFR3(5e4,[-1 1 -1 1 -1 1 .5 .5 .5] ,[pi/4 0 0])

DataGeneration5DOFSO3(5e4,[ 0 2\*pi 0 2\*pi 0 2\*pi pi/3 pi/3 pi/3] ,[.5 .5 .5])

DataGeneration5DOFSO3(5e4,[ 0 pi 0 pi 0 pi pi/3 pi/3 pi/3] ,[.5 .5 .5])

DataGeneration5DOFSO3(5e4,[ 0 2\*pi 0 2\*pi 0 2\*pi pi/2 pi/2 pi/2] ,[.5 .5 .5])

DataGeneration5DOFSO3(5e4,[0 2\*pi 0 2\*pi 0 2\*pi pi/3 pi/3 pi/3] ,[1 0 0])

DataGeneration6DOFR3(5e4,[-1 1 -1 1 -1 1 1 1 1] ,[0 0 0])

DataGeneration6DOFR3(5e4,[-1 1 -1 1 -1 1 .5 .5 .5] ,[0 0 0])

DataGeneration6DOFR3(5e4,[-1 1 -1 1 -1 1 .1 .1 .1] ,[0 0 0])

DataGeneration6DOFR3(5e4,[-1 1 -1 1 -1 1 .1 .1 .1] ,[pi/2 0 0])

DataGeneration6DOFR3(5e4,[-1 1 -1 1 -1 1 .1 .1 .1] ,[0 pi/2 0])

DataGeneration6DOFR3(5e4,[-1 1 -1 1 -1 1 .1 .1 .1] ,[0 0 pi/2])

DataGeneration6DOFR3(5e4,[-1 1 -1 1 -1 1 .1 .1 .1] ,[pi/4 pi/4 pi/4])

DataGeneration6DOFR3(5e4,[-1 1 -1 1 -1 1 .08 .08 .08] ,[0 0 0])

DataGeneration6DOFR3(5e4,[0 2 -1 1 -1 1 .1 .1 .1] ,[0 0 0])

DataGeneration6DOFR3(5e4,[-1 1 -1 1 -1 1 .1 .1 .1] ,[0 0 0])

DataGeneration6DOFR3(5e4,[-1 1 0 2 -1 1 .1 .1 .1] ,[0 0 0])

DataGeneration6DOFR3(5e4,[-1 1 -1 1 0 2 .1 .1 .1] ,[0 0 0])

DataGeneration6DOFR3(5e4,[-.5 1.5 -.5 1.5 -.5 1.5 .1 .1 .1] ,[0 0 0])

DataGeneration6DOFR3(5e4,[-1.5 1.5 -1.5 1.5 -1.5 1.5 .1 .1 .1] ,[0 0 0])

DataGeneration6DOFSO3(5e4,[ 0 2\*pi 0 2\*pi 0 2\*pi pi/10 pi/10 pi/10 ],[.5 .5 .5])

DataGeneration6DOFSO3(5e4, [ 0 2\*pi 0 2\*pi 0 2\*pi pi/3 pi/3 pi/3 ],[.5 .5 .5])

DataGeneration6DOFSO3(5e4, [ 0 pi 0 pi 0 pi pi/10 pi/10 pi/10 ],[.5 .5 .5])

DataGeneration6DOFSO3(5e4,[ 0 2\*pi 0 2\*pi 0 2\*pi pi/8 pi/8 pi/8 ],[1 0 0])

DataGeneration5DOFR3(5e4,[-1 1 -1 1 -1 1 .5 .5 .5] ,[0 0 0])

DataGeneration5DOFR3(5e4,[0 1 0 1 0 1 .5 .5 .5] ,[0 0 0])

DataGeneration5DOFR3(5e4,[-1 1 -1 1 -1 1 1 1 1] ,[0 0 0])

DataGeneration5DOFR3(5e4,[-1 1 -1 1 -1 1 .5 .5 .5] ,[pi/4 0 0])

DataGeneration5DOFSO3(5e4,[ 0 2\*pi 0 2\*pi 0 2\*pi pi/3 pi/3 pi/3] ,[.5 .5 .5])

DataGeneration5DOFSO3(5e4,[ 0 pi 0 pi 0 pi pi/3 pi/3 pi/3] ,[.5 .5 .5])

DataGeneration5DOFSO3(5e4,[ 0 2\*pi 0 2\*pi 0 2\*pi pi/2 pi/2 pi/2] ,[.5 .5 .5])

DataGeneration5DOFSO3(5e4,[0 2\*pi 0 2\*pi 0 2\*pi pi/3 pi/3 pi/3] ,[1 0 0])

DataGeneration6DOFR3(5e4,[-1 1 -1 1 -1 1 1 1 1] ,[0 0 0])

DataGeneration6DOFR3(5e4,[-1 1 -1 1 -1 1 .5 .5 .5] ,[0 0 0])

DataGeneration6DOFR3(5e4,[-1 1 -1 1 -1 1 .1 .1 .1] ,[0 0 0])

DataGeneration6DOFR3(5e4,[-1 1 -1 1 -1 1 .1 .1 .1] ,[pi/2 0 0])

DataGeneration6DOFR3(5e4,[-1 1 -1 1 -1 1 .1 .1 .1] ,[0 pi/2 0])

DataGeneration6DOFR3(5e4,[-1 1 -1 1 -1 1 .1 .1 .1] ,[0 0 pi/2])

DataGeneration6DOFR3(5e4,[-1 1 -1 1 -1 1 .1 .1 .1] ,[pi/4 pi/4 pi/4])

DataGeneration6DOFR3(5e4,[-1 1 -1 1 -1 1 .08 .08 .08] ,[0 0 0])

DataGeneration6DOFR3(5e4,[0 2 -1 1 -1 1 .1 .1 .1] ,[0 0 0])

DataGeneration6DOFR3(5e4,[-1 1 -1 1 -1 1 .1 .1 .1] ,[0 0 0])

DataGeneration6DOFR3(5e4,[-1 1 0 2 -1 1 .1 .1 .1] ,[0 0 0])

DataGeneration6DOFR3(5e4,[-1 1 -1 1 0 2 .1 .1 .1] ,[0 0 0])

DataGeneration6DOFR3(5e4,[-.5 1.5 -.5 1.5 -.5 1.5 .1 .1 .1] ,[0 0 0])

DataGeneration6DOFR3(5e4,[-1.5 1.5 -1.5 1.5 -1.5 1.5 .1 .1 .1] ,[0 0 0])

DataGeneration6DOFSO3(5e4,[ 0 2\*pi 0 2\*pi 0 2\*pi pi/10 pi/10 pi/10 ],[.5 .5 .5])

DataGeneration6DOFSO3(5e4, [ 0 2\*pi 0 2\*pi 0 2\*pi pi/3 pi/3 pi/3 ],[.5 .5 .5])

DataGeneration6DOFSO3(5e4, [ 0 pi 0 pi 0 pi pi/10 pi/10 pi/10 ],[.5 .5 .5])

DataGeneration6DOFSO3(5e4,[ 0 2\*pi 0 2\*pi 0 2\*pi pi/10 pi/10 pi/10 ],[1 0 0])

DataGeneration5DOFR3(5e4,[-1 1 -1 1 -1 1 .5 .5 .5] ,[0 0 0])

DataGeneration5DOFR3(5e4,[0 1 0 1 0 1 .5 .5 .5] ,[0 0 0])

DataGeneration5DOFR3(5e4,[-1 1 -1 1 -1 1 1 1 1] ,[0 0 0])

DataGeneration5DOFR3(5e4,[-1 1 -1 1 -1 1 .5 .5 .5] ,[pi/4 0 0])

DataGeneration5DOFSO3(5e4,[ 0 2\*pi 0 2\*pi 0 2\*pi pi/3 pi/3 pi/3] ,[.5 .5 .5])

DataGeneration5DOFSO3(5e4,[ 0 pi 0 pi 0 pi pi/3 pi/3 pi/3] ,[.5 .5 .5])

DataGeneration5DOFSO3(5e4,[ 0 2\*pi 0 2\*pi 0 2\*pi pi/2 pi/2 pi/2] ,[.5 .5 .5])

DataGeneration5DOFSO3(5e4,[ 0 2\*pi 0 2\*pi 0 2\*pi pi/3 pi/3 pi/3] ,[1 0 0])

DataGeneration6DOFR3(5e4,[-1 1 -1 1 -1 1 1 1 1] ,[0 0 0])

DataGeneration6DOFR3(5e4,[-1 1 -1 1 -1 1 .5 .5 .5] ,[0 0 0])

DataGeneration6DOFR3(5e4,[-1 1 -1 1 -1 1 .1 .1 .1] ,[0 0 0])

DataGeneration6DOFR3(5e4,[-1 1 -1 1 -1 1 .1 .1 .1] ,[pi/2 0 0])

DataGeneration6DOFR3(5e4,[-1 1 -1 1 -1 1 .1 .1 .1] ,[0 pi/2 0])

DataGeneration6DOFR3(5e4,[-1 1 -1 1 -1 1 .1 .1 .1] ,[0 0 pi/2])

DataGeneration6DOFR3(5e4,[-1 1 -1 1 -1 1 .1 .1 .1] ,[pi/4 pi/4 pi/4])

DataGeneration6DOFR3(5e4,[-1 1 -1 1 -1 1 .08 .08 .08] ,[0 0 0])

DataGeneration6DOFR3(5e4,[0 2 -1 1 -1 1 .1 .1 .1] ,[0 0 0])

DataGeneration6DOFR3(5e4,[-1 1 -1 1 -1 1 .1 .1 .1] ,[0 0 0])

DataGeneration6DOFR3(5e4,[-1 1 0 2 -1 1 .1 .1 .1] ,[0 0 0])

DataGeneration6DOFR3(5e4,[-1 1 -1 1 0 2 .1 .1 .1] ,[0 0 0])

DataGeneration6DOFR3(5e4,[-.5 1.5 -.5 1.5 -.5 1.5 .1 .1 .1] ,[0 0 0])

DataGeneration6DOFR3(5e4,[-1.5 1.5 -1.5 1.5 -1.5 1.5 .1 .1 .1] ,[0 0 0])

DataGeneration6DOFSO3(5e4,[ 0 2\*pi 0 2\*pi 0 2\*pi pi/10 pi/10 pi/10 ],[.5 .5 .5])

DataGeneration6DOFSO3(5e4, [ 0 2\*pi 0 2\*pi 0 2\*pi pi/3 pi/3 pi/3 ],[.5 .5 .5])

DataGeneration6DOFSO3(5e4, [ 0 pi 0 pi 0 pi pi/10 pi/10 pi/10 ],[.5 .5 .5])

DataGeneration6DOFSO3(5e4,[ 0 2\*pi 0 2\*pi 0 2\*pi pi/8 pi/8 pi/8 ],[1 0 0])

DataGeneration5DOFR3(5e4,[-1 1 -1 1 -1 1 .5 .5 .5] ,[0 0 0])

DataGeneration5DOFR3(5e4,[0 1 0 1 0 1 .5 .5 .5] ,[0 0 0])

DataGeneration5DOFR3(5e4,[-1 1 -1 1 -1 1 1 1 1] ,[0 0 0])

DataGeneration5DOFR3(5e4,[-1 1 -1 1 -1 1 .5 .5 .5] ,[pi/4 0 0])

DataGeneration5DOFSO3(5e4,[ 0 2\*pi 0 2\*pi 0 2\*pi pi/3 pi/3 pi/3] ,[.5 .5 .5])

DataGeneration5DOFSO3(5e4,[ 0 pi 0 pi 0 pi pi/3 pi/3 pi/3] ,[.5 .5 .5])

DataGeneration5DOFSO3(5e4,[ 0 2\*pi 0 2\*pi 0 2\*pi pi/2 pi/2 pi/2] ,[.5 .5 .5])

DataGeneration5DOFSO3(5e4,[0 2\*pi 0 2\*pi 0 2\*pi pi/3 pi/3 pi/3] ,[1 0 0])

%Final Test set

DataGeneration6DOFR3(1e3,[-1 1 -1 1 -1 1 1 1 1] ,[0 0 0])

DataGeneration6DOFR3(1e3,[-1 1 -1 1 -1 1 .5 .5 .5] ,[0 0 0])

DataGeneration6DOFR3(1e3,[-1 1 -1 1 -1 1 .1 .1 .1] ,[0 0 0])

DataGeneration6DOFR3(1e3,[-1 1 -1 1 -1 1 .1 .1 .1] ,[pi/2 0 0])

DataGeneration6DOFR3(1e3,[-1 1 -1 1 -1 1 .1 .1 .1] ,[0 pi/2 0])

DataGeneration6DOFR3(1e3,[-1 1 -1 1 -1 1 .1 .1 .1] ,[0 0 pi/2])

DataGeneration6DOFR3(1e3,[-1 1 -1 1 -1 1 .1 .1 .1] ,[pi/4 pi/4 pi/4])

DataGeneration6DOFR3(1e3,[-1 1 -1 1 -1 1 .08 .08 .08] ,[0 0 0])

DataGeneration6DOFR3(1e3,[0 2 -1 1 -1 1 .1 .1 .1] ,[0 0 0])

DataGeneration6DOFR3(1e3,[-1 1 -1 1 -1 1 .1 .1 .1] ,[0 0 0])

DataGeneration6DOFR3(1e3,[-1 1 0 2 -1 1 .1 .1 .1] ,[0 0 0])

DataGeneration6DOFR3(1e3,[-1 1 -1 1 0 2 .1 .1 .1] ,[0 0 0])

DataGeneration6DOFR3(1e3,[-.5 1.5 -.5 1.5 -.5 1.5 .1 .1 .1] ,[0 0 0])

DataGeneration6DOFR3(1e3,[-1.5 1.5 -1.5 1.5 -1.5 1.5 .1 .1 .1] ,[0 0 0])

DataGeneration6DOFSO3(1e3,[ 0 2\*pi 0 2\*pi 0 2\*pi pi/10 pi/10 pi/10 ],[.5 .5 .5])

DataGeneration6DOFSO3(1e3, [ 0 2\*pi 0 2\*pi 0 2\*pi pi/3 pi/3 pi/3 ],[.5 .5 .5])

DataGeneration6DOFSO3(1e3, [ 0 pi 0 pi 0 pi pi/10 pi/10 pi/10 ],[.5 .5 .5])

DataGeneration6DOFSO3(1e3,[ 0 2\*pi 0 2\*pi 0 2\*pi pi/8 pi/8 pi/8 ],[1 0 0])

DataGeneration5DOFR3(1e3,[-1 1 -1 1 -1 1 .5 .5 .5] ,[0 0 0])

DataGeneration5DOFR3(1e3,[0 1 0 1 0 1 .5 .5 .5] ,[0 0 0])

DataGeneration5DOFR3(1e3,[-1 1 -1 1 -1 1 1 1 1] ,[0 0 0])

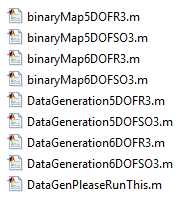
DataGeneration5DOFR3(1e3,[-1 1 -1 1 -1 1 .5 .5 .5] ,[pi/4 0 0])

DataGeneration5DOFSO3(1e3,[ 0 2\*pi 0 2\*pi 0 2\*pi pi/3 pi/3 pi/3] ,[.5 .5 .5])

DataGeneration5DOFSO3(1e3,[ 0 pi 0 pi 0 pi pi/3 pi/3 pi/3] ,[.5 .5 .5])

DataGeneration5DOFSO3(1e3,[ 0 2\*pi 0 2\*pi 0 2\*pi pi/2 pi/2 pi/2] ,[.5 .5 .5])

DataGeneration5DOFSO3(1e3,[0 2\*pi 0 2\*pi 0 2\*pi pi/3 pi/3 pi/3] ,[1 0 0])



Essay outline finished & section 1 done;

Date: 1/10/2018

Alexander Liao

Idea 3 (final)

Starting to collect test & training data from parallel computing workers

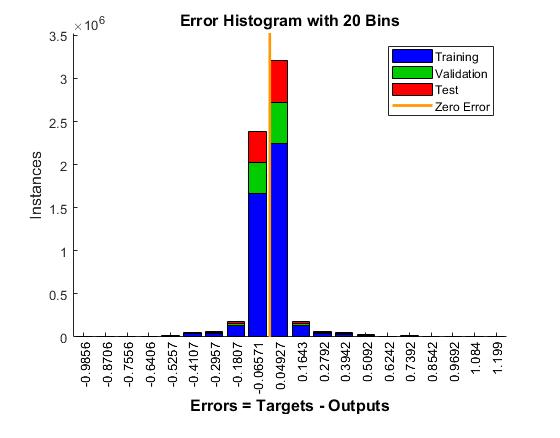
Date: 1/11/2018

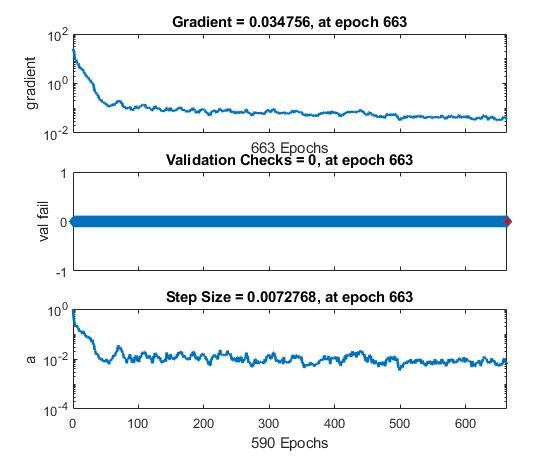
Alexander Liao

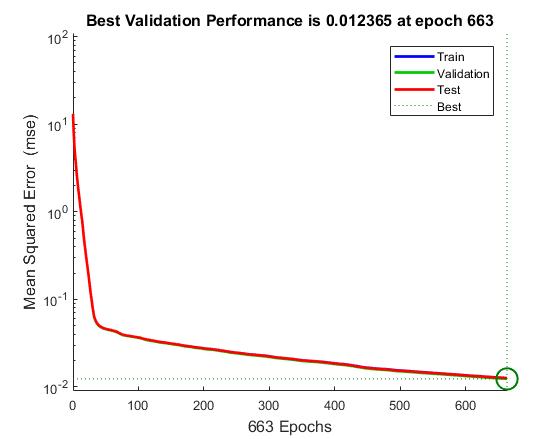
Idea 3 (final)

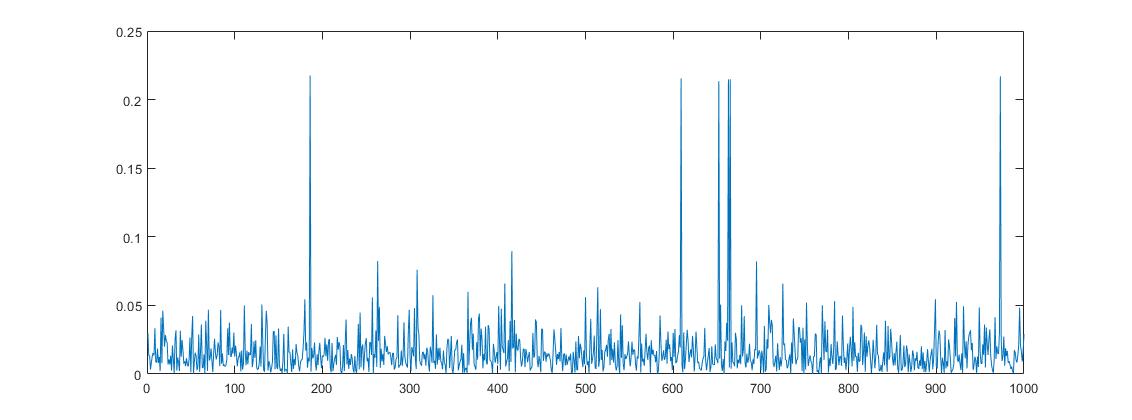
Trials on the final data

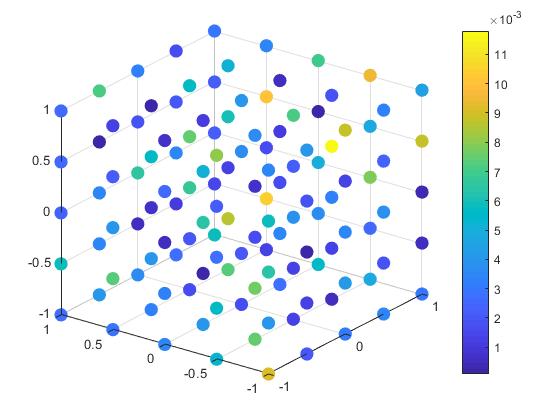
**Conjugate gradient with P/B researts: 125 pts; R^3; successful**







Average percentage error per sample:

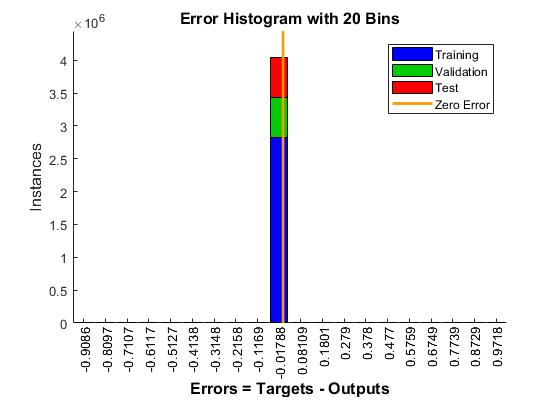
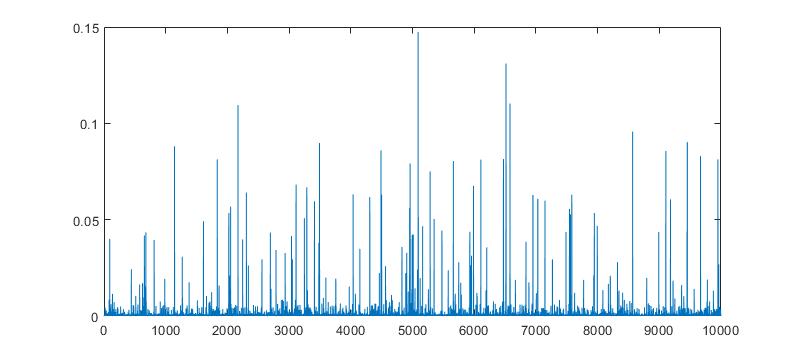
Average absolute error per sample:

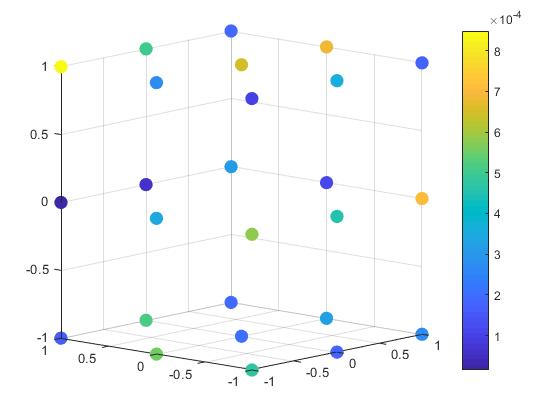
Additional trials on datasets

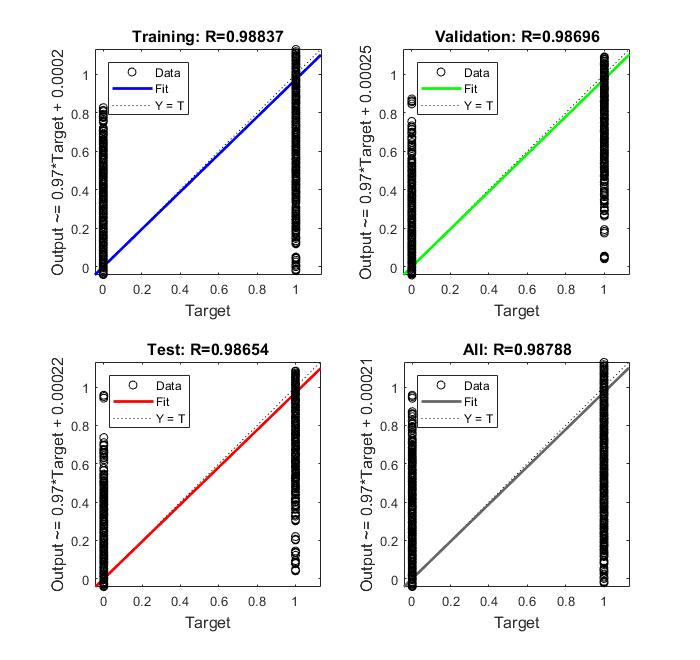
Date: 1/12/2018

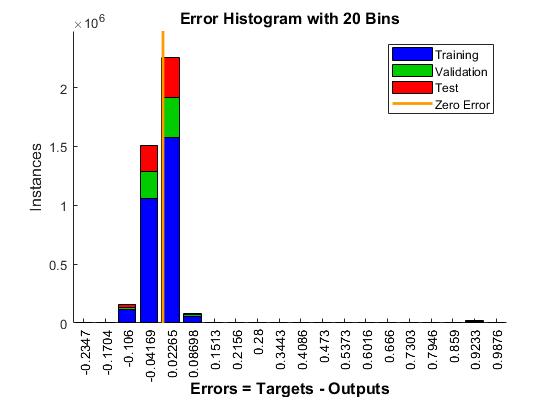
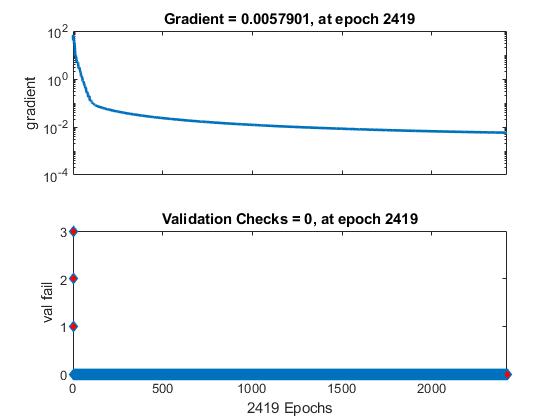
Alexander Liao

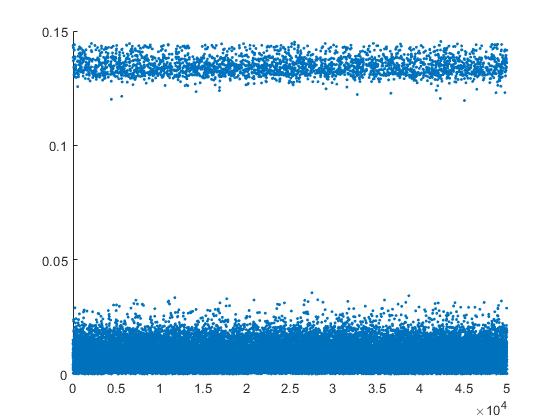
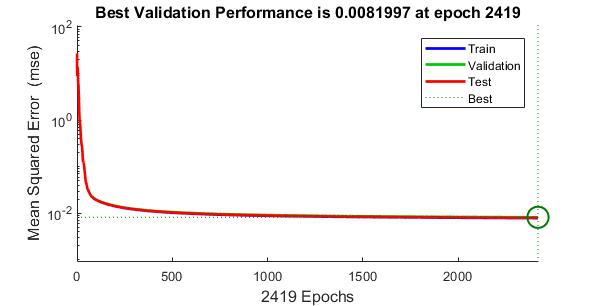
Idea 3 (final)

**Conjugate gradient with P/B researts: 27 pts; R^3; successful**

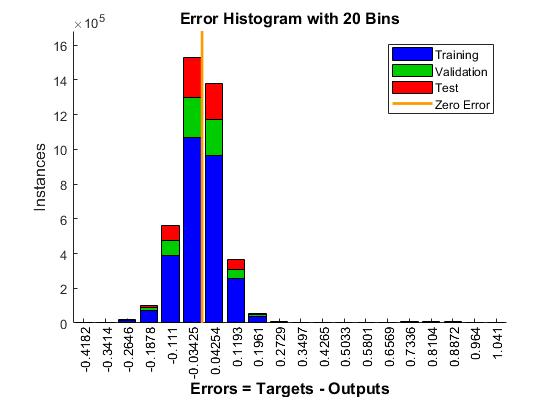




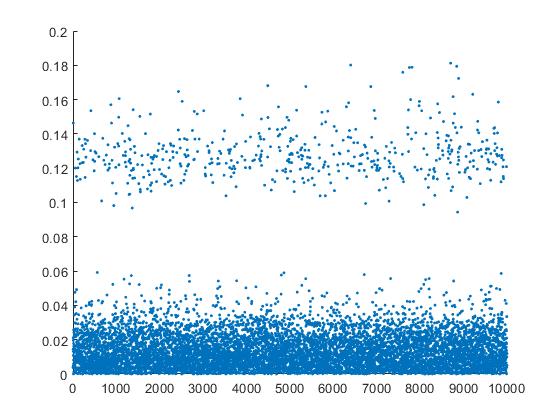
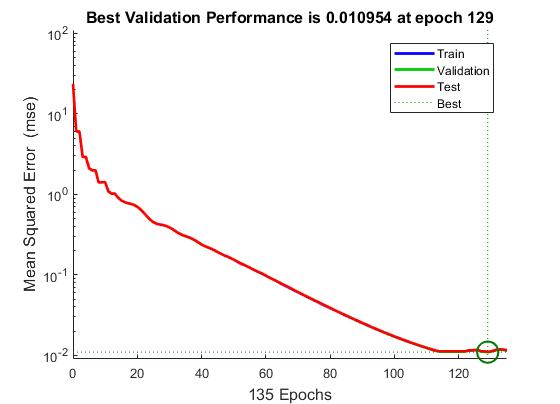
**Gradient descent with momentum: 27 pts; R^3; successful**

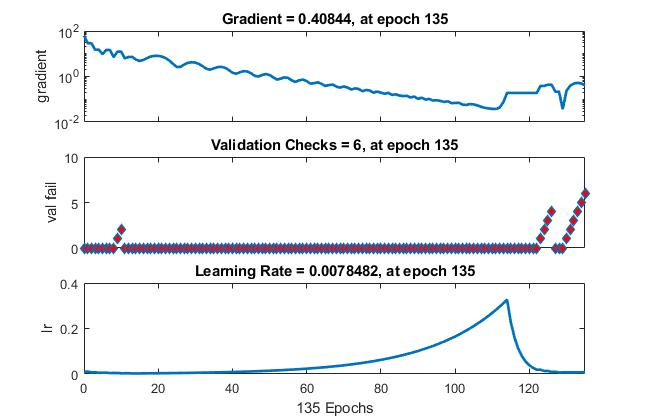
Average absolute error per prediction of samples

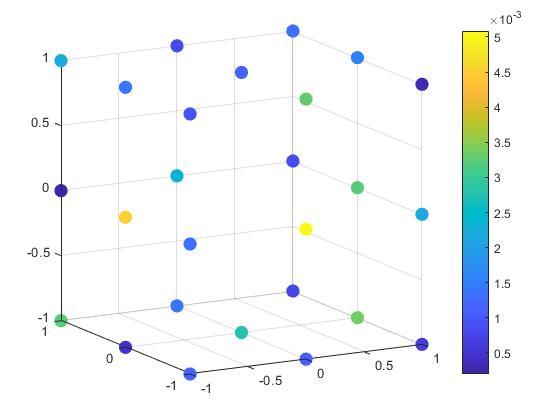
**Variable learning rate gradient descent: 27 pts; R^3; successful**



Average absolute error per prediction of samples







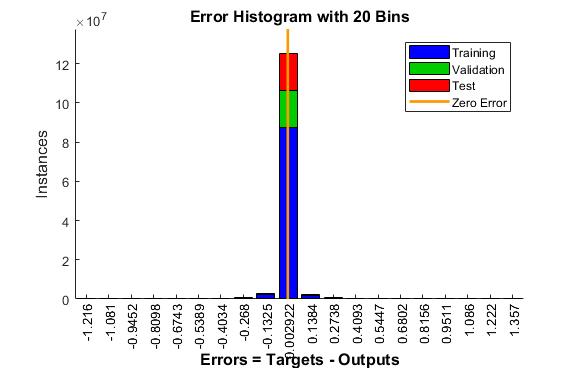
Additional trials on the datasets

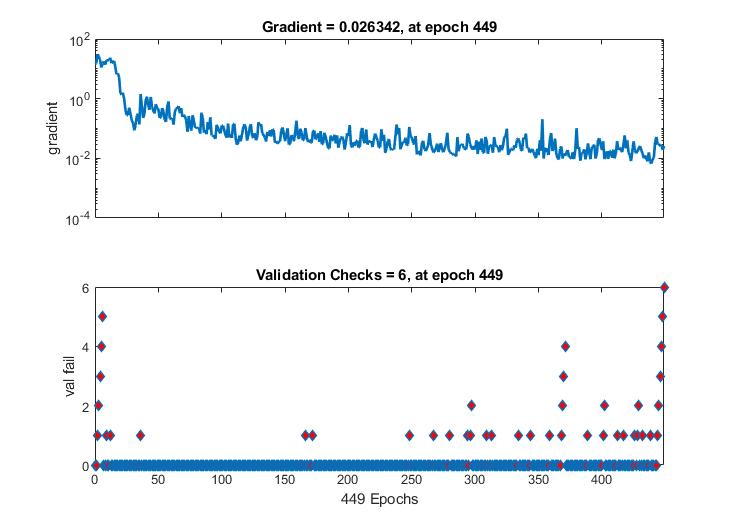
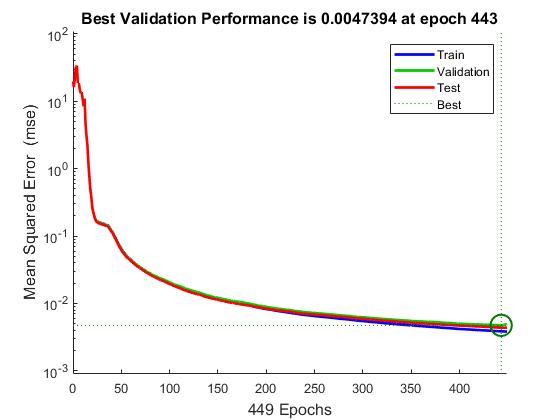
**Resilient backpropogation: 1331 pts; SO(3); successful**

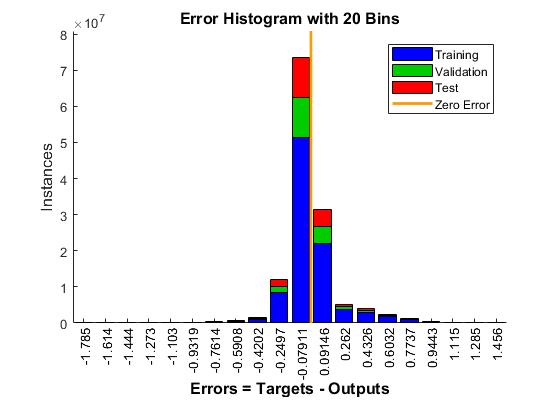
Date: 1/13/2018

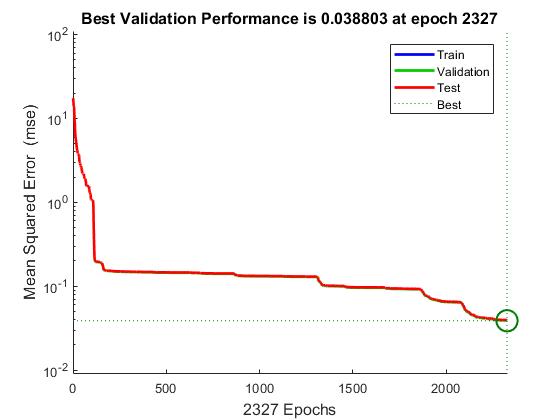
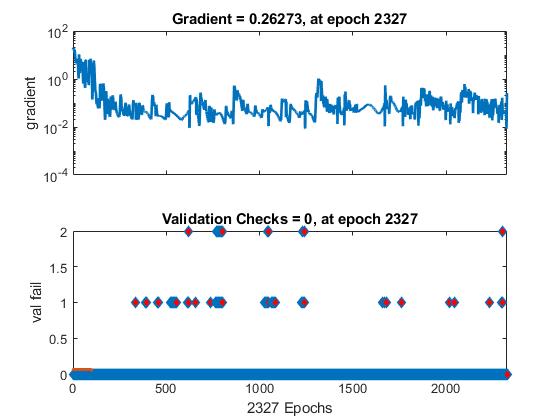
Alexander Liao

Idea 3 (final)

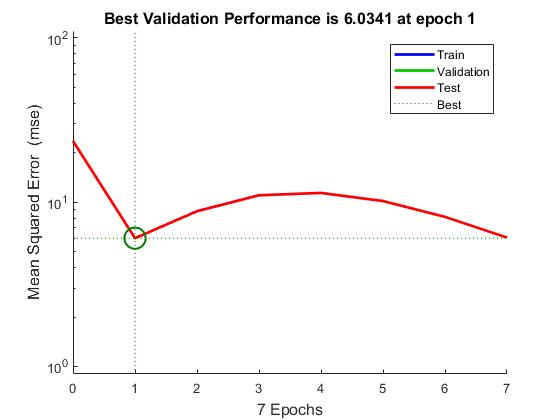
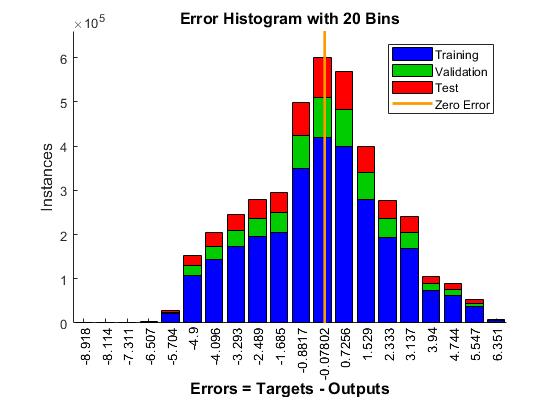
****

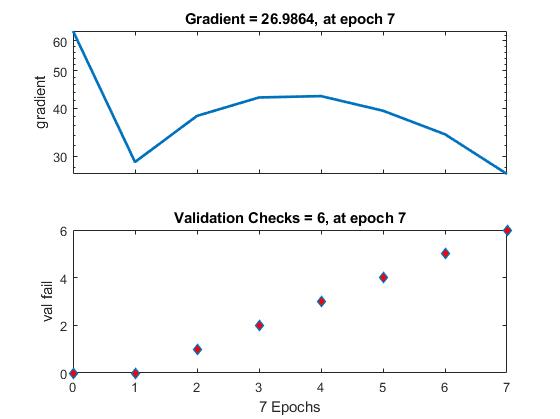
****

**One step secant: 1331 pts; SO(3);**

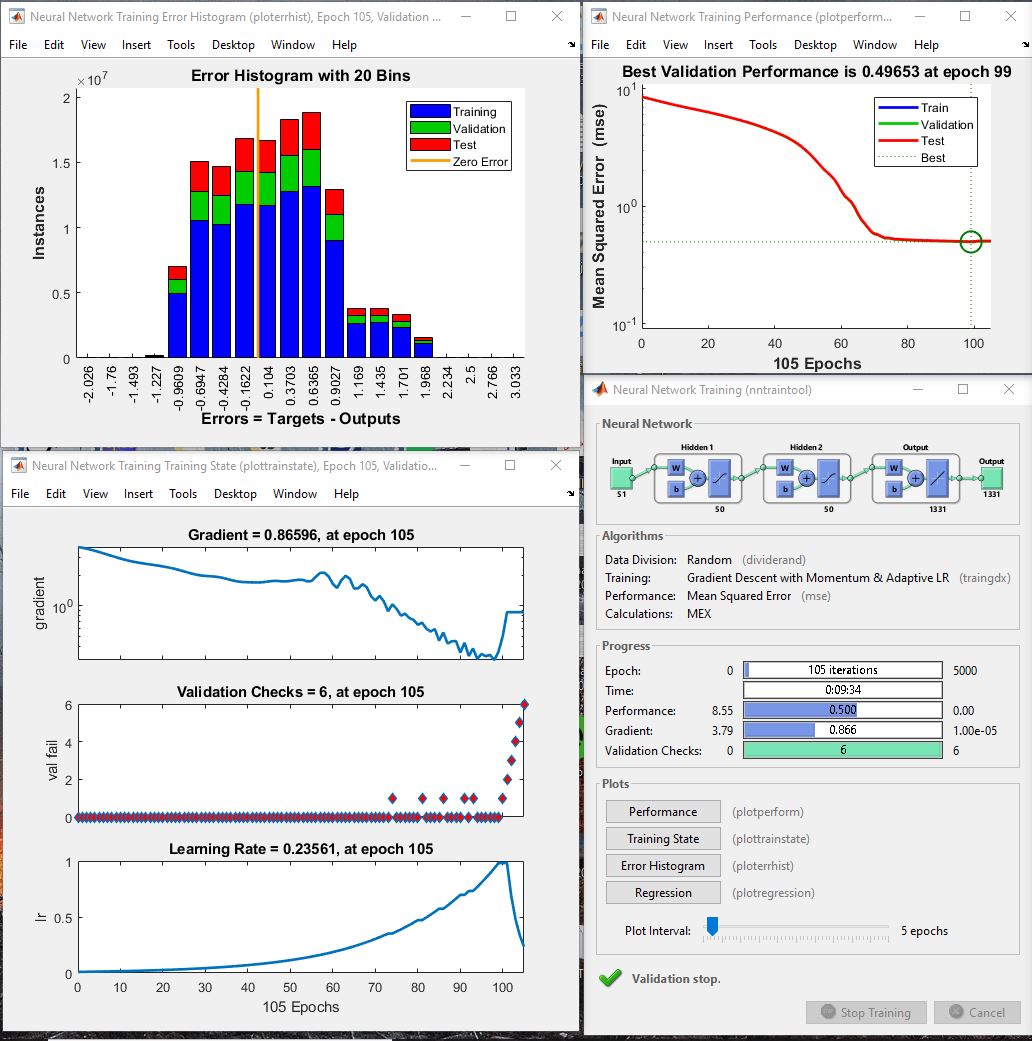


**Quasi-Newton backpropogation: 27 pts; R^3; failed**

****

****

**Variable learning rate gradient descent: 1331 pts; R^3; failed**

****

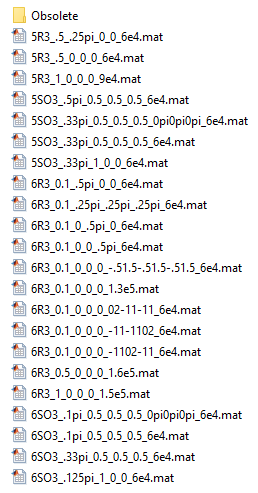
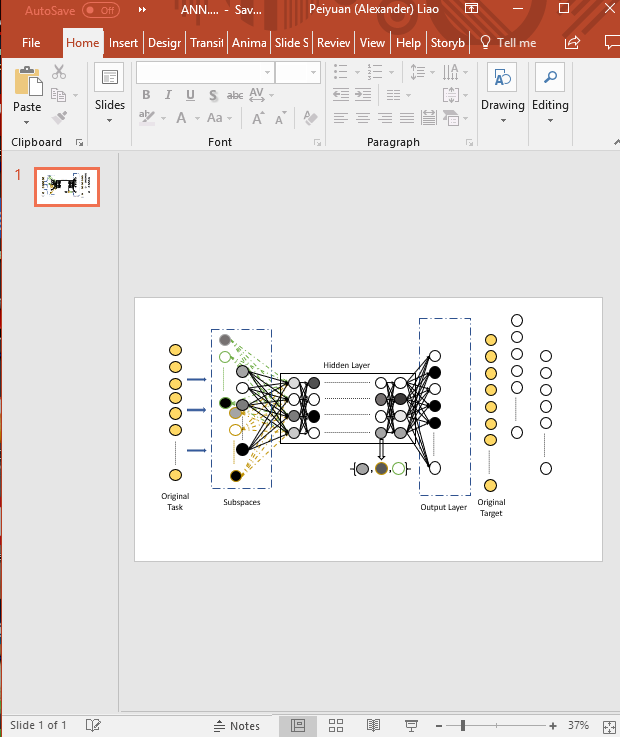
Train & test set complete

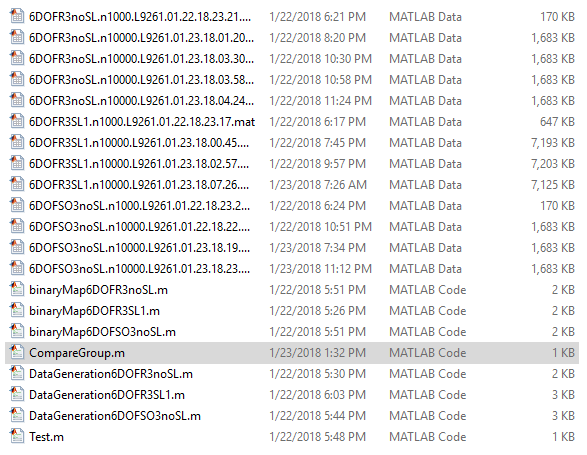
Date: 1/20/2018

Alexander Liao

Idea 3 (final)

Paper first draft complete (except data analysis)



Created compare group data generation script

D=[rand()/2 rand()/2 rand()/2 rand()/2 rand()/2 rand()/2];

A=[rand()/2 rand()/2 rand()/2 rand()/2 rand()/2 rand()/2];

alpha=2\*pi\*[ rand() rand() rand() rand() rand() rand()];

Date: 1/23/2018

Alexander Liao

Idea 3 (final)

1. Created scripts for model evaluation

Date: 1/27/2018

Alexander Liao

Idea 3 (final)

1. Script to learn the effect of stochasticity on the loos surface of the model

**Model Evaluation:**

load test27

resB=resB';

resM=resM';

rawNetData=net(resM);

roundedNetData=bR(rawNetData);

linearForm=[];

linearOriginal=[];

linearRaw=[];

[x,y]=size(roundedNetData);

resB=resB;

linearForm=zeros(1,x\*y);

linearOriginal=zeros(1,x\*y);

linearRaw=zeros(1,x\*y);

ct=1;

for i=1:x

for j=1:y

linearForm(ct)=roundedNetData(i,j);

linearOriginal(ct)=resB(i,j);

linearRaw(ct)=rawNetData(i,j);

ct=ct+1;

end

end

absoluteError=abs(linearRaw-linearOriginal);

truePos=0;

falseNeg=0;

falsePos=0;

[~,se]=size(linearForm);

for i=1:se

if and(linearForm(i)==1,linearOriginal(i)==1)

truePos=truePos+1;

end

if and(linearForm(i)==1,linearOriginal(i)==0)

falsePos=falsePos+1;

end

if and(linearForm(i)==0,linearOriginal(i)==1)

falseNeg=falseNeg+1;

end

end

precision=truePos/(truePos+falsePos);

recall=truePos/(truePos+falseNeg);

f\_score=2\*precision\*recall/(precision+recall);

function Mout = bR(Min)

[x,y]=size(Min);

Mout = zeros(x,y);

for i=1:x

for j=1:y

if (Min(i,j)>0.2)

Mout(i,j)=1;

end

end

end

end

**Stochasticity:**

gpu1 = gpuDevice(1);

%resM=resM';

%resB=resB';

z=zeros(10,10);

id1=1;

id2=1;

for epochs=100:-5:55

for batchNum=1:10

load train27

bSize=150000/batchNum;

bNum=batchNum;

net = feedforwardnet([100 50 100],'trainscg');

net.trainParam.epochs = epochs;

net.trainParam.max\_fail=12;

net.output.processFcns={'fixunknowns'};

net.input.processFcns={'mapminmax'};

%net.performFcn='mse';

for i=1:bNum

[net,tr] = train(net,datasample(resM,round(bSize))',datasample(resB,round(bSize))');

%finalTr{i}={tr.perf;tr.vperf;tr.tperf;tr.gradient};

end

load test27

resB=resB';

resM=resM';

rawNetData=net(resM);

roundedNetData=round(rawNetData);

linearForm=[];

linearOriginal=[];

linearRaw=[];

[x,y]=size(roundedNetData);

resB=resB;

linearForm=zeros(1,x\*y);

linearOriginal=zeros(1,x\*y);

linearRaw=zeros(1,x\*y);

ct=1;

for i=1:x

for j=1:y

linearForm(ct)=roundedNetData(i,j);

linearOriginal(ct)=resB(i,j);

linearRaw(ct)=rawNetData(i,j);

ct=ct+1;

end

end

absoluteError=abs(linearRaw-linearOriginal);

truePos=0;

falseNeg=0;

falsePos=0;

[~,se]=size(linearForm);

for i=1:se

if and(linearForm(i)==1,linearOriginal(i)==1)

truePos=truePos+1;

end

if and(linearForm(i)==1,linearOriginal(i)==0)

falsePos=falsePos+1;

end

if and(linearForm(i)==0,linearOriginal(i)==1)

falseNeg=falseNeg+1;

end

end

precision=truePos/(truePos+falsePos);

recall=truePos/(truePos+falseNeg);

f\_score=2\*precision\*recall/(precision+recall);

z1(id1,id2)=f\_score;

z2(id1,id2)=precision;

z3(id1,id2)=recall;

id2=id2+1;

end

id1=id1+1;

end

%genFunction(net);

precision=truePos/(truePos+falsePos);

recall=truePos/(truePos+falseNeg);

f\_score=2\*precision\*recall/(precision+recall);

function Mout = bR(Min)

[x,y]=size(Min);

Mout = zeros(x,y);

for i=1:x

for j=1:y

if (Min(i,j)>0.2)

Mout(i,j)=1;

end

end

end

end

rawNetData=net(resM);

roundedNetData=round(rawNetData);

linearForm=[];

linearOriginal=[];

linearRaw=[];

[x,y]=size(roundedNetData);

resB=resB;

linearForm=zeros(1,x\*y);

linearOriginal=zeros(1,x\*y);

linearRaw=zeros(1,x\*y);

ct=1;

for i=1:x

for j=1:y

linearForm(ct)=roundedNetData(i,j);

linearOriginal(ct)=resB(i,j);

linearRaw(ct)=rawNetData(i,j);

ct=ct+1;

end

end

absoluteError=abs(linearRaw-linearOriginal);

truePos=0;

falseNeg=0;

falsePos=0;

[~,se]=size(linearForm);

for i=1:se

if and(linearForm(i)==1,linearOriginal(i)==1)

truePos=truePos+1;

end

if and(linearForm(i)==1,linearOriginal(i)==0)

falsePos=falsePos+1;

end

if and(linearForm(i)==0,linearOriginal(i)==1)

falseNeg=falseNeg+1;

end

end

precision=truePos/(truePos+falsePos);

recall=truePos/(truePos+falseNeg);

f\_score=2\*precision\*recall/(precision+recall);

z1(id1,id2)=f\_score;

z2(id1,id2)=precision;

z3(id1,id2)=recall;

id2=id2+1;

end

id1=id1+1;

end

%genFunction(net);

Experimenting the delta=0.1 subspace model

Date: 1/28/2018

Alexander Liao

Idea 3 (final)

Sample code:

gpu1 = gpuDevice(1)

load train

resM=resM';

resB=resB';

bSize=5e4;

bNum=2;

for i=1:bNum

sample1=datasample(resM,bSize)';

sample2=datasample(resB,bSize)';

save(strcat('t',num2str(i),'.mat'),'sample1','sample2')

end

clear resB resM

net = feedforwardnet([100 50 50 50 50 50 50 100],'traincgp');

net.trainParam.epochs = 100;

net.trainParam.max\_fail=12;

net.output.processFcns={'fixunknowns'}

net.input.processFcns={'mapminmax'}

net.performFcn='mse'

for i=1:bNum

name=strcat('t',num2str(i),'.mat');

load(name)

[net,tr] = train(net,sample1,sample2);

finalTr{i}={tr.perf;tr.vperf;tr.tperf;tr.gradient};

end

%genFunction(net);

Experimenting on stochastic gradient descent.

Date: 1/29/2018

Alexander Liao

Idea 3 (final)

Abandoned due to unreliability

gpuDevice(1)

net = feedforwardnet([100 50 50 50 50 50 50 100],'trainscg');

net.trainParam.epochs = 250;

net.trainParam.max\_fail=12;

net.output.processFcns={'fixunknowns'}

net.input.processFcns={'fixunknowns'}

net.performFcn='mae'

net.divideParam.trainRatio=0.9;

net.divideParam.valRatio=0.05;

net.divideParam.testRatio=0.05;

ct=1

for l=1:4

for i=1:3

name=strcat('t',num2str(i),'.mat');

load(name)

[net,tr] = train(net,a(:,1:30000),b(:,1:30000));

finalTr{ct}={tr.perf;tr.vperf;tr.tperf;tr.gradient};

ct=ct+1;

end

l

end

clear a absoluteError b linearForm linearOriginal linearRaw

plot(1:max(size(gg)),gg,'lineWidth',2)

bSize=1000;

bNum=5000;

%load train

%resM=resM';

%resB=resB';

stopE=400;

%{

for i=501:5000

sample1=datasample(resM,bSize)';

sample2=datasample(resB,bSize)';

a=gpuArray(sample1);

b=gpuArray(sample2);

save(strcat('t',num2str(i),'.mat'),'a','b')

%ct=ct+bSize;

end

%}

gpu1 = gpuDevice(1)

clear resB resM sample1 sample2

net = feedforwardnet([100 100 100 100 100 100 200 400],'trainscg');

net.trainParam.epochs = 1;

net.output.processFcns={'mapminmax'}

net.input.processFcns={'fixunknowns'}

net.performFcn='mae'

lr=0.1;

finalTr=zeros(bNum,4);

%for l=1:3

net.trainParam.lr=lr;

%}

for i=1:bNum

if (i<stopE)

net.trainParam.lr=(1-(i/stopE))\*lr+(i/stopE)\*.01\*lr;

end

name=strcat('t',num2str(i),'.mat');

load(name)

[net,tr] = train(net,a,b,'useGPU','yes');

finalTr(i,:)=[tr.perf(2),tr.vperf(2),tr.tperf(2),tr.gradient(2)];

if mod(i,100)==0

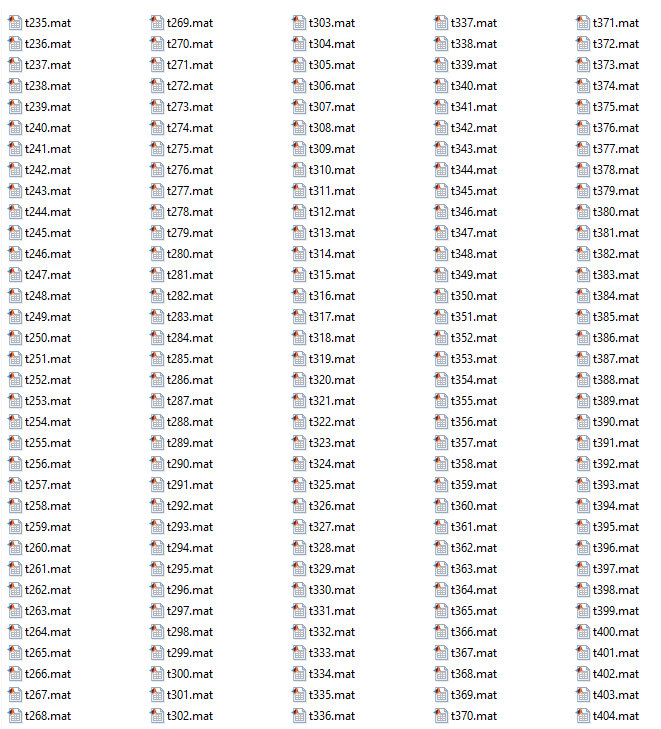
plot(1:max(max(size(finalTr))),finalTr(:,1)');

end

end

% l

%end

Training minibatches:

1. Script for run-time analysis

2/1/2018-2/4/2018

Wrapping-up on *Deep Learning*

2nd draft of paper

1. Delta=0.5 subspace test

Date: 2/5/2018

Alexander Liao

Idea 3 (final)

t1=zeros(1,200);

for i=1:200

tic;

rawNetData=net(resM(i,:)');

rND=bR(rawNetData,0.5);

a{i}=decoder(resM(i,:),rND);

t1(i)=toc;

end

plot(t1,'lineWidth',2)

t2=zeros(1,20);

for i=1:200

tic;

b{i}=CL(resM(i,:));

t2(i)=toc;

end

hold on

plot(t2,'lineWidth',2)

function Mout = bR(Min,n)

[x,y]=size(Min);

Mout = zeros(x,y);

for i=1:x

for j=1:y

if (Min(i,j)>n)

Mout(i,j)=1;

end

end

end

end

function P = CL(resM)

M=resM;

[~,s]=size(M);

D=M(1:6);

A=M(7:12);

alpha=M(13:18);

Coordinates=M(19:s);

stopPt=[];

dif=M(20)-M(19);

for i=20:s-1

if round((M(i+1)-M(i)),2)~=round(dif,2)

stopPt=horzcat(stopPt,i);

dif=M(i+2)-M(i+1);

end

end

if size(stopPt)==[1 1]

stopPt=horzcat(stopPt,18+(2\*(s-18)/3)+1);

elseif isempty(stopPt)

stopPt=[18+((s-18)/3)+1, 18+(2\*(s-18)/3)+1];

end

xArray=M(19:stopPt(1));

yArray=M(stopPt(1)+1:stopPt(2));

zArray=M(stopPt(2)+1:s);

[~,sizeX]=size(xArray);

[~,sizeY]=size(yArray);

[~,sizeZ]=size(zArray);

xmin=xArray(1);

xmax=xArray(sizeX);

ymin=yArray(1);

ymax=yArray(sizeY);

zmin=zArray(1);

zmax=zArray(sizeZ);

deltaX=xArray(2)-xArray(1);

deltaY=yArray(2)-yArray(1);

deltaZ=zArray(2)-zArray(1);

i=xmin;

j=ymin;

k=zmin;

c1=1;

c2=1;

c3=1;

l1=Link([0 D(1) A(1) alpha(1)]);

l2=Link([0 D(2) A(2) alpha(2)]);

l3=Link([0 D(3) A(3) alpha(3)]);

l4=Link([0 D(4) A(4) alpha(4)]);

l5=Link([0 D(5) A(5) alpha(5)]);

l6=Link([0 D(6) A(6) alpha(6)]);

m=SerialLink([l1,l2,l3,l4,l5,l6]);

H=cell( int32(((xmax-xmin)/deltaX)+1), int32(((ymax-ymin)/deltaY)+1), int32(((zmax-zmin)/deltaZ)+1) );

%The coordinates of the node is stored in a 3-D cell array

for k=zmin:deltaZ:zmax

c2=1;

for j=ymin:deltaY:ymax

c1=1;

for i=xmin:deltaX:xmax

H{c1, c2, c3}=[i,j,k];

c1=c1+1;

end

c2=c2+1;

end

c3=c3+1;

end

P=zeros(size(H));

for k=1:sizeZ

for j=1:sizeY

for i=1:sizeX

try

if all(not(isnan(m.ikine( trotx(angleX) \* troty(angleY) \* trotz(angleZ)\* transl(H{i,j,k}) ))))

% Analytical Inverse Kinematics Homogeneous Transformation Matrix

P(i,j,k)=1;

end

catch

end

end

end

end

end

function P= decoder(resM,resB1)

M=resM;

B=resB1;

[~,s]=size(M);

D=M(1:6);

A=M(7:12);

alpha=M(13:18);

Coordinates=M(19:s);

stopPt=[];

dif=M(20)-M(19);

for i=20:s-1

if round((M(i+1)-M(i)),2)~=round(dif,2)

stopPt=horzcat(stopPt,i);

dif=M(i+2)-M(i+1);

end

end

if size(stopPt)==[1 1]

stopPt=horzcat(stopPt,18+(2\*(s-18)/3)+1);

elseif isempty(stopPt)

stopPt=[18+((s-18)/3)+1, 18+(2\*(s-18)/3)+1];

end

xArray=M(19:stopPt(1));

yArray=M(stopPt(1)+1:stopPt(2));

zArray=M(stopPt(2)+1:s);

[~,sizeX]=size(xArray);

[~,sizeY]=size(yArray);

[~,sizeZ]=size(zArray);

xmin=xArray(1);

xmax=xArray(sizeX);

ymin=yArray(1);

ymax=yArray(sizeY);

zmin=zArray(1);

zmax=zArray(sizeZ);

deltaX=xArray(2)-xArray(1);

deltaY=yArray(2)-yArray(1);

deltaZ=zArray(2)-zArray(1);

i=xmin;

j=ymin;

k=zmin;

c1=1;

c2=1;

c3=1;

index=1;

for k=zmin:deltaZ:zmax

c2=1;

for j=ymin:deltaY:ymax

c1=1;

for i=xmin:deltaX:xmax

H{c1, c2, c3}=[i,j,k];

c1=c1+1;

end

c2=c2+1;

end

c3=c3+1;

end

[sizeX,sizeY,sizeZ]=size(H);

P=zeros(size(H));

for k=1:sizeZ

for j=1:sizeY

for i=1:sizeX

P(i,j,k)=B(index);

index=index+1;

end

end

end

end

[~,s]=size(M);

D=M(1:6);

A=M(7:12);

alpha=M(13:18);

Coordinates=M(19:s);

stopPt=[];

dif=M(20)-M(19);

for i=20:s-1

if round((M(i+1)-M(i)),2)~=round(dif,2)

stopPt=horzcat(stopPt,i);

dif=M(i+2)-M(i+1);

end

end

if size(stopPt)==[1 1]

stopPt=horzcat(stopPt,18+(2\*(s-18)/3)+1);

elseif isempty(stopPt)

stopPt=[18+((s-18)/3)+1, 18+(2\*(s-18)/3)+1];

end

xArray=M(19:stopPt(1));

yArray=M(stopPt(1)+1:stopPt(2));

zArray=M(stopPt(2)+1:s);

[~,sizeX]=size(xArray);

[~,sizeY]=size(yArray);

[~,sizeZ]=size(zArray);

xmin=xArray(1);

xmax=xArray(sizeX);

ymin=yArray(1);

ymax=yArray(sizeY);

zmin=zArray(1);

zmax=zArray(sizeZ);

deltaX=xArray(2)-xArray(1);

deltaY=yArray(2)-yArray(1);

deltaZ=zArray(2)-zArray(1);

i=xmin;

j=ymin;

k=zmin;

c1=1;

c2=1;

c3=1;

l1=Link([0 D(1) A(1) alpha(1)]);

l2=Link([0 D(2) A(2) alpha(2)]);

l3=Link([0 D(3) A(3) alpha(3)]);

l4=Link([0 D(4) A(4) alpha(4)]);

l5=Link([0 D(5) A(5) alpha(5)]);

l6=Link([0 D(6) A(6) alpha(6)]);

m=SerialLink([l1,l2,l3,l4,l5,l6]);

H=cell( int32(((xmax-xmin)/deltaX)+1), int32(((ymax-ymin)/deltaY)+1), int32(((zmax-zmin)/deltaZ)+1) );

%The coordinates of the node is stored in a 3-D cell array

for k=zmin:deltaZ:zmax

c2=1;

for j=ymin:deltaY:ymax

c1=1;

for i=xmin:deltaX:xmax

H{c1, c2, c3}=[i,j,k];

c1=c1+1;

end

c2=c2+1;

end

c3=c3+1;

end

P=zeros(size(H));

for k=1:sizeZ

for j=1:sizeY

for i=1:sizeX

try

if all(not(isnan(m.ikine( trotx(angleX) \* troty(angleY) \* trotz(angleZ)\* transl(H{i,j,k}) ))))

% Analytical Inverse Kinematics Homogeneous Transformation Matrix

P(i,j,k)=1;

end

catch

end

end

end

end

end

function P= decoder(resM,resB1)

M=resM;

B=resB1;

[~,s]=size(M);

D=M(1:6);

A=M(7:12);

alpha=M(13:18);

Coordinates=M(19:s);

stopPt=[];

dif=M(20)-M(19);

for i=20:s-1

if round((M(i+1)-M(i)),2)~=round(dif,2)

stopPt=horzcat(stopPt,i);

dif=M(i+2)-M(i+1);

end

end

if size(stopPt)==[1 1]

stopPt=horzcat(stopPt,18+(2\*(s-18)/3)+1);

elseif isempty(stopPt)

stopPt=[18+((s-18)/3)+1, 18+(2\*(s-18)/3)+1];

end

xArray=M(19:stopPt(1));

yArray=M(stopPt(1)+1:stopPt(2));

zArray=M(stopPt(2)+1:s);

[~,sizeX]=size(xArray);

[~,sizeY]=size(yArray);

[~,sizeZ]=size(zArray);

xmin=xArray(1);

xmax=xArray(sizeX);

ymin=yArray(1);

ymax=yArray(sizeY);

zmin=zArray(1);

zmax=zArray(sizeZ);

deltaX=xArray(2)-xArray(1);

deltaY=yArray(2)-yArray(1);

deltaZ=zArray(2)-zArray(1);

i=xmin;

j=ymin;

k=zmin;

c1=1;

c2=1;

c3=1;

index=1;

for k=zmin:deltaZ:zmax

c2=1;

for j=ymin:deltaY:ymax

c1=1;

for i=xmin:deltaX:xmax

H{c1, c2, c3}=[i,j,k];

c1=c1+1;

end

c2=c2+1;

end

c3=c3+1;

end

[sizeX,sizeY,sizeZ]=size(H);

P=zeros(size(H));

for k=1:sizeZ

for j=1:sizeY

for i=1:sizeX

P(i,j,k)=B(index);

index=index+1;

end

end

end

end

H=cell( int32(((xmax-xmin)/deltaX)+1), int32(((ymax-ymin)/deltaY)+1), int32(((zmax-zmin)/deltaZ)+1) );

%The coordinates of the node is stored in a 3-D cell array

for k=zmin:deltaZ:zmax

c2=1;

for j=ymin:deltaY:ymax

c1=1;

for i=xmin:deltaX:xmax

H{c1, c2, c3}=[i,j,k];

c1=c1+1;

end

c2=c2+1;

end

c3=c3+1;

end

P=zeros(size(H));

for k=1:sizeZ

for j=1:sizeY

for i=1:sizeX

try

if all(not(isnan(m.ikine( trotx(angleX) \* troty(angleY) \* trotz(angleZ)\* transl(H{i,j,k}) ))))

% Analytical Inverse Kinematics Homogeneous Transformation Matrix

P(i,j,k)=1;

end

catch

end

end

end

end

end

function P= decoder(resM,resB1)

M=resM;

B=resB1;

[~,s]=size(M);

D=M(1:6);

A=M(7:12);

alpha=M(13:18);

Coordinates=M(19:s);

stopPt=[];

dif=M(20)-M(19);

for i=20:s-1

if round((M(i+1)-M(i)),2)~=round(dif,2)

stopPt=horzcat(stopPt,i);

dif=M(i+2)-M(i+1);

end

end

if size(stopPt)==[1 1]

stopPt=horzcat(stopPt,18+(2\*(s-18)/3)+1);

elseif isempty(stopPt)

stopPt=[18+((s-18)/3)+1, 18+(2\*(s-18)/3)+1];

end

xArray=M(19:stopPt(1));

yArray=M(stopPt(1)+1:stopPt(2));

zArray=M(stopPt(2)+1:s);

[~,sizeX]=size(xArray);

[~,sizeY]=size(yArray);

[~,sizeZ]=size(zArray);

xmin=xArray(1);

xmax=xArray(sizeX);

ymin=yArray(1);

ymax=yArray(sizeY);

zmin=zArray(1);

zmax=zArray(sizeZ);

deltaX=xArray(2)-xArray(1);

deltaY=yArray(2)-yArray(1);

deltaZ=zArray(2)-zArray(1);

i=xmin;

j=ymin;

k=zmin;

c1=1;

c2=1;

c3=1;

index=1;

for k=zmin:deltaZ:zmax

c2=1;

for j=ymin:deltaY:ymax

c1=1;

for i=xmin:deltaX:xmax

H{c1, c2, c3}=[i,j,k];

c1=c1+1;

end

c2=c2+1;

end

c3=c3+1;

end

[sizeX,sizeY,sizeZ]=size(H);

P=zeros(size(H));

for k=1:sizeZ

for j=1:sizeY

for i=1:sizeX

P(i,j,k)=B(index);

index=index+1;

end

end

end

end

if size(stopPt)==[1 1]

stopPt=horzcat(stopPt,18+(2\*(s-18)/3)+1);

elseif isempty(stopPt)

stopPt=[18+((s-18)/3)+1, 18+(2\*(s-18)/3)+1];

end

xArray=M(19:stopPt(1));

yArray=M(stopPt(1)+1:stopPt(2));

zArray=M(stopPt(2)+1:s);

[~,sizeX]=size(xArray);

[~,sizeY]=size(yArray);

[~,sizeZ]=size(zArray);

xmin=xArray(1);

xmax=xArray(sizeX);

ymin=yArray(1);

ymax=yArray(sizeY);

zmin=zArray(1);

zmax=zArray(sizeZ);

deltaX=xArray(2)-xArray(1);

deltaY=yArray(2)-yArray(1);

deltaZ=zArray(2)-zArray(1);

i=xmin;

j=ymin;

k=zmin;

c1=1;

c2=1;

c3=1;

index=1;

for k=zmin:deltaZ:zmax

c2=1;

for j=ymin:deltaY:ymax

c1=1;

for i=xmin:deltaX:xmax

H{c1, c2, c3}=[i,j,k];

c1=c1+1;

end

c2=c2+1;

end

c3=c3+1;

end

[sizeX,sizeY,sizeZ]=size(H);

P=zeros(size(H));

for k=1:sizeZ

for j=1:sizeY

for i=1:sizeX

P(i,j,k)=B(index);

index=index+1;

end

end

end

end

end

Experimental attempt of generating samples without subspace embedding

Date: 2/6/2018

Alexander Liao

Idea 3 (final)

load train

resB=resB';

resM=resM';

gpuDevice(1)

net = feedforwardnet([250 250 250 250 300 300 300 500],'trainscg');

net.trainParam.epochs = 300;

net.trainParam.max\_fail=6;

net.output.processFcns={'fixunknowns'}

net.input.processFcns={'fixunknowns'}

net.performFcn='mse'

[net,tr] = train(net,resM,resB);

2/7/2018-2/8/2018

Writing a library for subspace learning

Setting up a Github repository for research codes

%orientation = [ 0 0 0];

% Angular displacement x-axis y z (in radians)

angleX=rand()\*2\*pi;

angleY=rand()\*2\*pi;

angleZ=rand()\*2\*pi;

Scripts to compute f-score for a model

Date: 2/9/2018

Alexander Liao

Idea 3 (final)

load test

resB=resB';

resM=resM';

resB=resB(1323\*3+1:1323\*4,:);

f\_final=[];

for filter=-1:0.1:1

rawNetData=net(resM);

roundedNetData=bR(rawNetData,filter);

linearForm=[];

linearOriginal=[];

linearRaw=[];

[x,y]=size(roundedNetData);

linearForm=zeros(1,x\*y);

linearOriginal=zeros(1,x\*y);

linearRaw=zeros(1,x\*y);

ct=1;

for i=1:x

for j=1:y

linearForm(ct)=roundedNetData(i,j);

linearOriginal(ct)=resB(i,j);

linearRaw(ct)=rawNetData(i,j);

ct=ct+1;

end

end

absoluteError=abs(linearRaw-linearOriginal);

truePos=0;

falseNeg=0;

falsePos=0;

[~,se]=size(linearForm);

for i=1:se

if and(linearForm(i)==1,linearOriginal(i)==1)

truePos=truePos+1;

end

if and(linearForm(i)==1,linearOriginal(i)==0)

falsePos=falsePos+1;

end

if and(linearForm(i)==0,linearOriginal(i)==1)

falseNeg=falseNeg+1;

end

end

end

precision=truePos/(truePos+falsePos);

recall=truePos/(truePos+falseNeg);

f\_score=2\*precision\*recall/(precision+recall);

f\_final=horzcat(f\_final,f\_score);

end

l=[]

for i=1:5

l=horzcat(l,cell2mat(finalTr{i}(4)))

end

function Mout = bR(Min,n)

[x,y]=size(Min);

Mout = zeros(x,y);

for i=1:x

for j=1:y

if (Min(i,j)>n)

Mout(i,j)=1;

end

end

end

end

precision=truePos/(truePos+falsePos);

recall=truePos/(truePos+falseNeg);

f\_score=2\*precision\*recall/(precision+recall);

f\_final=horzcat(f\_final,f\_score);

end

l=[]

for i=1:5

l=horzcat(l,cell2mat(finalTr{i}(4)))

end

function Mout = bR(Min,n)

[x,y]=size(Min);

Mout = zeros(x,y);

for i=1:x

for j=1:y

if (Min(i,j)>n)

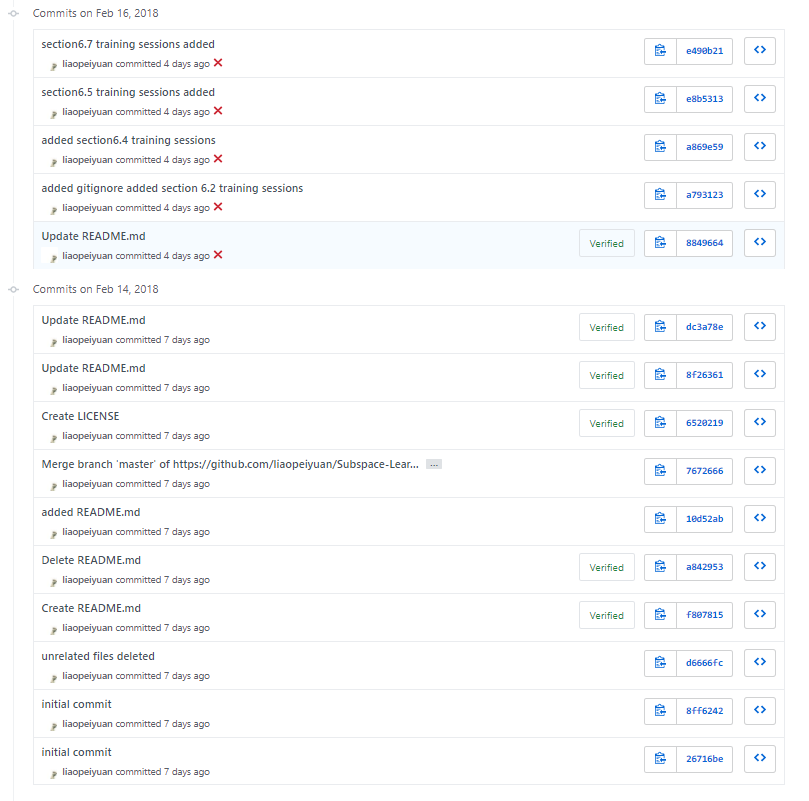
Mout(i,j)=1;

end

end

end

end



2/14/2018-2/20/2018

Creting files on Github

