SU(2) LQCD

Sean Hannaford

Jun 19, 2020

CONTENTS:

su2.**calcPlaq**(*U*, *V*, *mups*)
Calculates the average value of the plaquettes about all points in the lattice

Parameters

- **U** (*array_like*) – Array containing the gaugefields for every point on the lattice
- **V** (*int*) – The volume of the lattice, which is equivalent to the number of points on the lattice
- **mups** (*array_like*) – The mups array. This array is used as shorthand for taking a step forwards in the mu'th direction from the U0i'th point

Returns The average value of the plaquettes about the whole lattice

Return type numpy.float64

su2.**calcU_i**(*U*, *V*, *La*, *mups*)
Calculates the average values of the spacial links in the lattice

Parameters

- **U** (*array_like*) – Array containing the gaugefields for every point on the lattice
- **V** (*int*) – The volume of the lattice, which is equivalent to the number of points on the lattice
- **mups** (*array_like*) – The mups array. This array is used as shorthand for taking a step forwards in the mu'th direction from the U0i'th point

Returns The average value of the spacial links in the lattice

Return type numpy.float64

su2.**calcU_t**(*U*, *V*, *mups*)
Calculates the average values of the time links in the lattice

Parameters

- **U** (*array_like*) – Array containing the gaugefields for every point on the lattice
- **V** (*int*) – The volume of the lattice, which is equivalent to the number of points on the lattice
- **mups** (*array_like*) – The mups array. This array is used as shorthand for taking a step forwards in the mu'th direction from the U0i'th point

Returns The average value of the time links in the lattice

Return type numpy.float64

su2.**compare**(*mat1*, *mat2*)
Prints any dissimilar elements between two matrices

Parameters

- **mat1** (*array_like*) – A matrix
- **mat2** (*array_like*) – A matrix

Returns Compares corresponding elements between two matrices and prints any values which are not exactly equal.

Return type void

su2.**corr**(*x*)
Writes out the correlator to a file

Parameters **x** (*array_like*) – The inverse

su2.**cstart**()
Returns 2x2 identity matrix

Returns 2x2 Identity matrix written in the convention of a real-valued SU(2) matrix

Return type numpy.ndarray

su2.**dag**(*U*)
Gives the hermitian conjugate of a matrix writen in the real-valued representation of an SU(2) matrix

Parameters **u** (*array_like*) – Real-valued matrix representaion of an SU(2) matrix

Returns Hermitian conjugate of the input written as a real-valued representation of an SU(2) matrix

Return type numpy.ndarray

su2.**dagger**(*u*)
Gives the hermitian conjugate of a matrix

Parameters **u** (*array_like*) – Matrix representing a gauge field

Returns Hermitian conjugate of the input

Return type numpy.ndarray

su2.**det**(*UU*)
Returns the determinant of the matrix

Parameters **UU** (*array_like*) – SU(2) matrix writen in real-valued form

Returns Determinant of the input matrix

Return type numpy.float64

su2.**dim**(*La*)
Returns the dimensions of the array in a dictionary

Parameters **La** (*array_like*) – Array where each element describes the length of one dimension of the lattice ([x,y,z,t])

Returns Dictionary containing the lattice dimensions

Return type dict

su2.**getElement**(*D*, *La*, *point_1*, *point_2*, *a*, *alpha*, *b*, *beta*)
input the matrix, the dimensions of the latice, the initial and final point of the particle, and the other spin and field stuff, and get the associated element of the matrix point_1, point_2 range from 0 to [max index] - 1 a, b range from 0 to 1 alpha, beta range from 0 to 3

su2.**getIndex**(*La*, *point_1*)
Returns the index of the full Dirac matrix which corresponds to the element associated with a particular point on the lattice

Parameters

- **La** (*array_like*) – Array where each element describes the length of one dimension of the lattice ([x,y,z,t])
- **point** (*array_like*) – Array containing the spacetime coordinates of a position on the lattice written as [x,y,z,t]

Returns This function operates very similarly to p2i

Return type int

su2.**getMups**(*V*, *numdim*, *La*)
Returns the mups array

Parameters

- **V** (*int*) – The volume of the lattice, which is equivalent to the number of points on the lattice
- **numdim** (*int*) – Number of dimensions of the lattice
- **La** (*array_like*) – Array where each element describes the length of one dimension of the lattice ([x,y,z,t])

Returns The output is a V x numdim matrix array which can be used as shorthand when calling elements from the gaugefield array. Specifically, this array can be used in functions which involve stepping up or down between points on the lattice. In the output matrix array, the ith array corresponds to the ith point on the lattice, and the elements of that array are the indexes of the adjacent points. For example, the [i,mu] element in the output array is the index of point on the lattice corresponding to the point one step in the mu'th direction.

Return type numpy.ndarray

su2.**getTime**(*a*, *b*)
Displays the elalpsed time in hr:min:sec format

Parameters

- **a** (*float*) – The current time given by the time() function in the standard time module
- **b** (*float*) – The current time given by the time() function in the standard time module

Returns Calculates the time elapsed between when a and b were initialized and print the elapsed time in hr:min:sec format. Can be used to find the time elapsed by a section of code if a is initialized before said section and b is intialized immediately after

Return type string

su2.**getstaple**(*U*, *U0i*, *mups*, *mdns*, *mu*)
Returns the value of the staple

Parameters

- **U** (*array_like*) – Array containing the gaugefields for every point on the lattice
- **U0i** (*int*) – Lattice point index of the starting point on the lattice for the calculation.
- **mups** (*array_like*) – The mups array. This array is used as shorthand for taking a step forwards in the mu'th direction from the U0i'th point
- **mdns** (*array_like*) – The mdns array. This array is used as shorthand for taking a step backwards in the mu'th direction from the U0i'th point
- **mu** (*int*) – Index corresponding to one of the directions on the lattice: 0:x, 1:y, 2:z, 3:t

Returns Returns the staple starting at the U0i'th point

Return type numpy.ndarray

su2.**hstart**()
Returns a random complex 2x2 matrix written in real-valued form

Returns 2x2 matrix written in real-valued form. Elements are assigned to random values between -1 and 1

Return type numpy.ndarray

su2.**i2p**(*ind*, *La*)

Takes the index of a point on the lattice and returns the spacetime position of that point

Parameters

- **ind** (*int*) – As the elements of the Dirac matrix are themselves 8x8 matrices, i is the index of the first row/colum of the elements in the Dirac matrix which pertain to a particular spacetime position
- **La** (*array_like*) – Array where each element describes the length of one dimension of the lattice ([x,y,z,t])

Returns The position on the lattice which corresponds to the input index. The position is written as [x,y,z,t]

Return type numpy.ndarray

su2.**initD**(*row*, *col*, *dat*, *mu*, *U*, *r*, *m*, *n*, *pbc*)

Generates the data needed to make an 8x8 sparse submatrix containing the kinetic terms (including time) of the dirac matrix.

Parameters

- **row** (*list*) – list containing row indices for the non-zero elements of the dirac matrix. Is used to generate the dirac matrix as a sparse matrix.
- **col** (*list*) – list containing column indices for the non-zero elements of the dirac matrix. Is used to generate the dirac matrix as a sparse matrix.
- **dat** (*list*) – list containing the values of the non-zero elements of the dirac matrix. Is used to generate the dirac matrix as a sparse matrix.
- **mu** (*int*) – Index corresponding to one of the directions on the lattice: 0:x, 1:y, 2:z, 3:t
- **U** (*array_like*) – Array containing the gaugefields for every point on the lattice
- **r** (*double*) – the value of the wilson term
- **m** (*int*) – row index of the Dirac matrix which, when used in conjunction with n, will locate the first diagonal element of the submatrix
- **n** (*int*) – column index of the Dirac matrix which, when used in conjunction with m, will locate the first diagonal element of the submatrix
- **pbc** (*bool*) – boolean variable indicating whethere the particle has triggered periodic boundary conditions. Is true when the particle hit a boundary and had to be moved to the opposite end of the lattice, and false otherwise.

Returns Appends the row, col, and dat lists with the data needed to construct an 8x8 submatrix which will be part of the larger dirac matrix. The submatrix is the kronecker product of the gamma matrix corresponding to the mu direction and the gaugefield connecting the lattice points given by su2.i2p(m//8,La) and su2.i2p(n//8,La). Returns error codes if an error occurs or 0 otherwise, and the number of elements initialized.

Return type int, int

su2.**initDeo**(*row*, *col*, *dat*, *mu*, *U*, *r*, *m*, *n*, *pbc*)

Generates the data needed to make an 8x8 sparse, even/odd precoditioned submatrix containing the kinetic terms (including time) of the dirac matrix.

Parameters

- **row** (*list*) – list containing row indices for the non-zero elements of the dirac matrix. Is used to generate the dirac matrix as a sparse matrix.

- **col** (*list*) – list containing column indices for the non-zero elements of the dirac matrix. Is used to generate the dirac matrix as a sparse matrix.
- **dat** (*list*) – list containing the values of the non-zero elements of the dirac matrix. Is used to generate the dirac matrix as a sparse matrix.
- **mu** (*int*) – Index corresponding to one of the directions on the lattice: 0:x, 1:y, 2:z, 3:t
- **U** (*array_like*) – Array containing the gaugefields for every point on the lattice
- **r** (*double*) – the value of the wilson term
- **m** (*int*) – row index of the Dirac matrix which, when used in conjunction with n, will locate the first diagonal element of the submatrix
- **n** (*int*) – column index of the Dirac matrix which, when used in conjunction with m, will locate the first diagonal element of the submatrix
- **pbc** (*bool*) – boolean variable indicating whethere the particle has triggered periodic boundary conditions. Is true when the particle hit a boundary and had to be moved to the opposite end of the lattice, and false otherwise.

Returns This function takes advantage of even/odd preconditioning. Since the Dirac matrix can be broken into mass terms and terms which either connect "even" points with "odd" points or vice versa, the Dirac matrix can be formulated as three smaller matrices which are half the order of the full matrix. So, this fucntion appends the row, col, and dat lists with the data needed to construct an 8x8 submatrix which will be part of the larger even-odd or odd-even precoditioned matrix. The submatrix is the kronecker product of the gamma matrix corresponding to the mu direction and the gaugefield connecting the lattice points given by su2.i2p(m//8,La) and su2.i2p(n//8,La). Returns error codes if an error occurs or 0 otherwise, and the number of elements initialized.

Return type int, int

su2.**link**(*U*, *U0i*, *mups*, *mu*)

Returns the trace of the link between two points

Parameters

- **U** (*array_like*) – Array containing the gaugefields for every point on the lattice
- **U0i** (*int*) – Lattice point index of the starting point on the lattice for the calculation.
- **mups** (*array_like*) – The mups array. This array is used as shorthand for taking a step forwards in the mu'th direction from the U0i'th point
- **mu** (*int*) – Index corresponding to one of the directions on the lattice: 0:x, 1:y, 2:z, 3:t

Returns The value of the link between the point at U0i and the point one step in the mu'th direction

Return type numpy.float64

su2.**masseo**(*row*, *dat*, *i*, *j*, *m*, *r*)

Generates the data needed to make an 8x8 sparse submatrix containing the mass terms of the dirac matrix

Parameters

- **row** (*list*) – list containing row indices for the non-zero elements of the dirac matrix. Is used to generate the dirac matrix as a sparse matrix.
- **dat** (*list*) – list containing the values of the non-zero elements of the dirac matrix. Is used to generate the dirac matrix as a sparse matrix.
- **i** (*int*) – row index of the Dirac matrix which, when used in conjunction with j, will locate the first diagonal element of the submatrix

- **j** (*int*) – column index of the Dirac matrix which, when used in conjunction with i, will locate the first diagonal element of the submatrix
- **m** (*double*) – the mass of the particle
- **r** (*double*) – the value of the wilson term

Returns Appends the row and dat lists with the data needed to construct an 8x8 submatrix which will be part of the larger dirac matrix. All mass terms should be along the diagonal of each submatrix as well as the larger dirac matrix.

Return type void

su2.**mdowni**(*ind*, *mu*, *La*)

Decrement a position in the mu'th direction, looping if needed

Parameters

- **ind** (*int*) – As the elements of the Dirac matrix are themselves 8x8 matrices, i is the index of the first row/colum of the elements in the Dirac matrix which pertain to a particular spacetime position
- **mu** (*int*) – Index corresponding to one of the directions on the lattice: 0:x, 1:y, 2:z, 3:t
- **La** (*array_like*) – Array where each element describes the length of one dimension of the lattice ([x,y,z,t])

Returns The function deccrements a step in the mu'th direction, if the boundary is met it then loops around to the other side of the lattice. The return value is the index of the new point on the lattice.

Return type numpy.int64

su2.**mult**(*U1*, *U2*)

Multiplies two SU(2) matrices written in the real-valued representation

Parameters

- **U1** (*array_like*) – Real-valued matrix representaion of an SU(2) gauge field
- **U2** (*array_like*) – Real-valued matrix representaion of an SU(2) gauge field

Returns The product of the two input arrays written as a real-valued matrix representaion of an SU(2) matrix

Return type numpy.ndarray

su2.**mupi**(*ind*, *mu*, *La*)

Increment a position in the mu'th direction, looping if needed

Parameters

- **ind** (*int*) – the index of a point on the lattice
- **mu** (*int*) – Index corresponding to one of the directions on the lattice: 0:x, 1:y, 2:z, 3:t
- **La** (*array_like*) – Array where each element describes the length of one dimension of the lattice ([x,y,z,t])

Returns The function increments a step in the mu'th direction, if the boundary is met it then loops around to the other side of the lattice. The return value is the index of the new point on the lattice.

Return type numpy.int64

su2.**p2i**(*point*, *La*)
Takes the array describing a point in the spacetime lattice ([x,y,z,t] notation) and returns the index of that point

Parameters

- **point** (*array_like*) – Array containing the spacetime coordinates of a position on the lattice written as [x,y,z,t]
- **La** (*array_like*) – Array where each element describes the length of one dimension of the lattice ([x,y,z,t])

Returns The index of the input point

Return type int

su2.**parity**(*pt*)
Returns the parity of a point on the lattice

Parameters **pt** (*array_like*) – Point on the lattice, written as [x,y,z,t]

Returns Returns 1 if the point has even parity, and 0 if it has odd parity

Return type numpy.int64

su2.**plaq**(*U*, *U0i*, *mups*, *mu*, *nu*)
Compute the plaquette

U [array_like] Array containing the gaugefields for every point on the lattice

U0i [int] Lattice point index of the starting point on the lattice for the calculation.

mups [array_like] The mups array. This array is used as shorthand for taking a step forwards in the mu'th direction from the U0i'th point

mu [int] Index corresponding to one of the directions on the lattice: 0:x, 1:y, 2:z, 3:t

nu [int] Index corresponding to another direction on the lattice: 0:x, 1:y, 2:z, 3:t.

Returns The value of the plaquette

Return type numpy.float64

su2.**showMc**(*mat*, *rc*)
Prints the submatrices along the leftmost column of the full Dirac matrix

Parameters

- **mat** (*array_like*) – The Dirac matrix
- **rc** (*int*) – The order of the Dirac matrix

Returns Prints 8x8 submatrices that exist along the leftmost section of the Dirac matrix.

Return type void

su2.**showMr**(*mat*, *rc*)
Prints the submatrices along the top row of the full Dirac matrix

Parameters

- **mat** (*array_like*) – The Dirac matrix
- **rc** (*int*) – The order of the Dirac matrix

Returns Prints 8x8 submatrices that exist along the top of the Dirac matrix.

Return type void

su2.**showU**(*U*, *mu*, *i*)
Returns the gauge field at the i'th lattice point and in the mu'th direction written as matrix writen in the real-valued representation of an SU(2) matrix

Returns

- **U** (*array_like*) – Real-valued matrix representaion of an SU(2) gauge field
- **mu** (*int*) – Index corresponding to one of the directions on the lattice: 0:x, 1:y, 2:z, 3:t
- **i** (*int*) – row index of the Dirac matrix which, when used in conjunction with j, will locate the first diagonal element of the submatrix

Returns The gauge field which corresponds to the mu'th direction on the i'th point on the lattice. It is a 2x2 matrix written in the real-valued representation

Return type numpy.ndarray

su2.**staple**(*U*, *U0i*, *mups*, *mdns*, *mu*, *nu*, *signnu*)
Compute the staple in the mu-nu plane

Parameters

- **U** (*array_like*) – Array containing the gaugefields for every point on the lattice
- **U0i** (*int*) – Lattice point index of the starting point on the lattice for the calculation.
- **mups** (*array_like*) – The mups array. This array is used as shorthand for taking a step forwards in the mu'th direction from the U0i'th point
- **mdns** (*array_like*) – The mdns array. This array is used as shorthand for taking a step backwards in the mu'th direction from the U0i'th point
- **mu** (*int*) – Index corresponding to one of the directions on the lattice: 0:x, 1:y, 2:z, 3:t
- **nu** (*int*) – Index corresponding to one of the directions on the lattice: 0:x, 1:y, 2:z, 3:t. Must be different than mu
- **signnu** (*int*) – Equal to either 1 or -1. Dictates if the staple is calulcuted forwards (+1) or in reverse (-1).

Returns Returns the staple starting at the U0i'th point

Return type numpy.ndarray

su2.**tr**(*UU*)
Return the trace of a matrix

Parameters **UU** (*array_like*) – SU(2) matrix writen in real-valued form

Returns The trace of the input matrix

Return type numpy.float64

su2.**update**(*UU*)
Make a random SU(2) matrix near the identity

Parameters **UU** (*array_like*) – SU(2) matrix written in real-valued form

Returns Updated version of the input matrix with a slight random modification. Matrix is near the identity and written in real-valued form

Return type numpy.ndarray

su2.**vol**(*La*)
Takes array of dimensions as input, returns volume

Parameters **La** (*array_like*) – Array where each element describes the length of one dimension of the lattice ([x,y,z,t])

Returns The volume of the lattice, which is equivalent to the number of points on the lattice

Return type int

dirac_generator.**gen_dirac**(*U*, *m*, *spaceLength*, *timeLength*, *r=1.0*)

Generates a set of preconditioned matrices which comprise a Dirac matrix

Returns

- **De** (*array_like*) – A preconditioned matrix containing the elements which connect an "even" point on the lattice with an "odd" point. Is half the order of the full Dirac matrix
- **Do** (*array_like*) – A preconditioned matrix containing the elements which connect an "odd" point on the lattice with an "even" point. All elements are kinetic terms and De is half the order of the full Dirac matrix
- **Dm** (*array_like*) – A preconditioned matrix containing the elements containing the mass terms. Is half the order of the full Dirac matrix.

dirac_generator.**invert_dirac**(*De*, *Do*, *Dm*, *sizeofinverse=8*)

Takes in the even/odd preconditioned Dirac matrices and finds the inverse of the full Dirac matrix

Parameters

- **De** (*array_like*) – A preconditioned matrix containing the elements which connect an "even" point on the lattice with an "odd" point. Is half the order of the full Dirac matrix
- **Do** (*array_like*) – A preconditioned matrix containing the elements which connect an "odd" point on the lattice with an "even" point. All elements are kinetic terms and De is half the order of the full Dirac matrix
- **Dm** (*array_like*) – A preconditioned matrix containing the elements containing the mass terms. Is half the order of the full Dirac matrix.

Returns The inverse of the Dirac matrix. By default only the first 8 rows of the inverse are calculated.

Return type numpy.ndarray

Other Parameters **sizeofinverse** (*int*) – The number of rows of the inverse that will be calculated.

field_gen.**lattice**(*bS*, *spaceLength*, *timeLength*, *M=1000*, *Mlink=10*, *Msep=10*, *Mtherm=100*, *bRange=0.0*, *bI=0.1*, *hotStart=False*, *makePlot=True*)

Generates a lattice or set of lattices utilizing SU(2) gauge fields

Parameters

- **bS** (*float*) – The value of beta that will be used to generate a lattice. If a set of lattices generated over a range of betas is desired, then bS is the first value of beta evaluated over the range.
- **spaceLength** (*int*) – The length of the three spacial dimensions of the lattice
- **timeLength** (*int*) – The length of the time dimension of the lattice

Returns **th** – Numpy array where the fist element is the average value of the plaquettes of the lattice, and the second element is the lattice. While the function only returns the final update of the lattice, the lattice is written out into the configs directory each time a measurement is made.

Return type numpy.ndarray

Other Parameters

- **M** (*int*) – The number of times the lattice is updated. An update of the lattice is performed by iterating through each lattice point and each link extending from that point and updating those links. If enough updates are performed on the lattice it will converge to some configuration. Set ot 1000 by default.
- **Mlink** (*int*) – The number of times each link is updated per lattice update. A link is updated by creating a random SU(2) matrix near the identity, then calculating the action of the new matrix. If the action is decreased or by some random probability the new matrix will be assigned to the link, otherwise the link does not change. This process is repeated Mlink times. Set to 10 by default
- **Msep** (*int*) – The separation of measurements. While the lattice is being updated M times, a measurement of the average value of the plaquettes in the lattice will be taken every Msep updates. Each time a measurement is taken, the lattice will also be written out to a pickled file. (E.g. if Msep = 10, measurements will be taken every 10 updates). Set to 10 by default
- **Mtherm** (*int*) – The "thermalization" of the lattice. Indicates the number of lattice updates that need to be performed before any measurements of the average value of the plaquettes in the lattice. Set to 100 by default.
- **bRange** (*float*) – The difference between the greatest and smallest value of beta that will be used to generate a lattice. A bRange of 0 means only one value of beta will be used to make a lattice. Set to 0.0 by default.
- **bI** (*float*) – If it is desired to generate a set of lattices using a range of betas, bI is how much beta will increment between values of beta. Set to 0.1 by default
- **hotStart** (*bool*) – Setting this to True will initialize the fields of the lattice with a hot start (randomly generated values) rather than the default cold start (SU(2) matrices set to the identity). Set to False by default.
- **makePlot** (*bool*) – When True, plots of the plaquette vs. updates will be generated. No plots generated when makePlot is set to False. Set to True by default.

CHAPTER

ONE

INDICES AND TABLES

- genindex
- modindex
- search

PYTHON MODULE INDEX

d

f

s

INDEX