Descriptions of each function

- The numbers on the graph in the 'Graph of functions in variational_MPS_algorithm.pdf' file indicate the order in which the main function variational_ground_state_MPS calls each other function.
- In green boxes are the functions which are part of the while loop of the main algorithm
- The functions get_identity_MPO, get_Ising_MPO, inner_product_MPS, get_spin_half_expectation_value and get_spin_half_MPO are not part of the main algorithm to find the ground state and its energy.
- In the description of the functions below, the names of the variables will be underlined and highlighted.

1) initialize_MPS:

Creates a random mps as a vector of arrays with 3 indices holding complex 64-bit floats.

2) gauge_MPS!:

Will mutate the mps and put it in left or right canonical form using gauge_site on each element of the mps vector. This function also calls the contraction function which is used throughout the algorithm from various other functions and its functionality is to contract two tensors together along some given indices.

contraction:

Contracts two tensors together along some given indices and returns the result of the contraction as an array of complex floats.

gauge_site:

Puts a site into left or right canonical form by using the singular value decomposition function svd of LinearAlgebra (https://docs.julialang.org/en/v1/stdlib/LinearAlgebra/#LinearAlgebra.svd).

3) initialize_L_R_states:

Creates a vector called states that holds the partial contractions of the <mps|mpo|mps> diagram. For example, states at element i will hold the contracted <mps|mpo|mps> diagram up to site i on the lattice. The first and last elements of states are dummy 1x1x1 tensors with value 1 that are used when we are trying to update site 1 or the last site of the mps respectively.

Now we enter the while loop which sweeps the lattice from left to right and from right to left updating each site's mps tensor by picking the one that will minimise the effective Hamiltonian. The while loop will stop if we reach the maximum number of iterations we chose to do or if the fractional change in energy after a full right and left sweep is smaller than a given accuracy.

4) get_updated_site:

Calculates the tensor on a given site that will minimise the energy of the effective Hamiltonian. To get the effective Hamiltonian, this function calls get_Heff. To get the optimized site tensor it calls eigs of Arpack (https://arpack.julialinearalgebra.org/latest/index.html#Arpack.eigs-Tuple{Any}).

get_Heff:

Contracts the left part of the effective Hamiltonian diagram with the mpo tensor at the site index we are trying to optimize and update, and then it contracts the result with the right part of the effective Hamiltonian to obtain the effective Hamiltonian. It then reshapes the result to return a matrix of two indices. It also returns the dimensions of each index that were merged into forming the two indices of the matrix to be used by get_updated_site to reshape the optimized site mps tensor from a vector of one index to a 3-tensor.

5) gauge_site:

This function is called again to gauge the site we just updated and maintain the mps into the proper mixed canonical form. To the left of the site we are going to update next we need the mps to be in left canonical form and to the right of the site we are going to update next we need the mps to be in right canonical form, forming the proper mixed canonical form.

6) update_states!:

Mutates the states vector created by the initialize_L_R_states functions so as to incorporate the updated and optimized mps 3-tensor that came from get_updated_site. The update we need to do on states is to computer the result <mps|mpo|mps> on the site index we just updated and then contract the result with the element of states at index i-1 or i+1 depending on whether the sweep is going towards the right or left respectively.

7) contractions:

This function is called just before we break the while loop and it contracts the first site of mps with the resulting matrices of the singular value decomposition after we gauge site 2 when the sweep that goes from right to left finishes. This contraction is needed to return a normalized mps such that <mps|mps> = 1.

get_identity_MPO:

This function returns the identity operator as an MPO (matrix product operator) representation that can act on a state represented with an MPS (matrix product state).

get_Ising_MPO:

This function returns the 1D Ising Hamiltonian as an MPO (matrix product operator) representation that can act on a state represented with an MPS (matrix product state). The Hamiltonian is given by

$$\sum_{\langle i,j\rangle} J\hat{Z}_i\hat{Z}_j + \sum_i g_x\hat{X}_i + \sum_i g_z\hat{Z}_i$$

where the capital letters represent the Pauli spin matrices.