SRG and NCSM

SRG evolution in three-nucleon space
NCSM solution for the triton

Initialization

- Here are some constants that will be needed later-on as well as some optional settings for the graphics output. Please evaluate these cells when you start working with this notebook.

```
hbarc = 197.327053;                    (* [MeV fm] *)
MN0 = (938.27231 + 939.56563) / 2;     (* average nucleon mass [MeV] *)
MN = MN0 / hbarc^2 ;
(* average nucleon mass in units [1/(MeV fm^2)] *)
```

```
SetOptions[ArrayPlot, ColorFunction → "SunsetColors"];
SetOptions[DensityPlot, ColorFunction → "SunsetColors"];
SetOptions[ListDensityPlot, ColorFunction → "SunsetColors"];
SetOptions[ListLinePlot, PlotMarkers → Automatic,
  Frame → True, Axes → False, AspectRatio → 0.8];
```

Initial Hamiltonian

- In this project we solve the quantum many-body problem for the ground state of the triton (^3H) using an SRG-transformed chiral Hamiltonian in the No-Core Shell Model.
- In a first step, we prepare the initial Hamiltonian, which is given in a matrix representation using a three-body Jacobi-coordinate harmonic-oscillator basis. Those matrix elements are provided via external data files.
- The initial Hamiltonian is composed of the kinetic energy T and the interaction V, which in our case is the most recent chiral two- plus three-nucleon interaction used also in the lecture on ab initio methods.
- In order to allow for numerical calculations, these two operators have to be represented in a suitable basis, i.e., the matrix elements of the operators for all combinations of two basis states are used as input. The basis of choice for the present application in the three-nucleon system is a antisymmetrized three-body Jacobi-coordinate harmonic oscillator basis. This basis never appears explicitly in this notebook (because it's quite complicated), but it was used to compute the matrix elements which are provided in the data files.
- First we have to set the working directory to the place, where you have put the matrix element files. You might have to change the following statement accordingly.

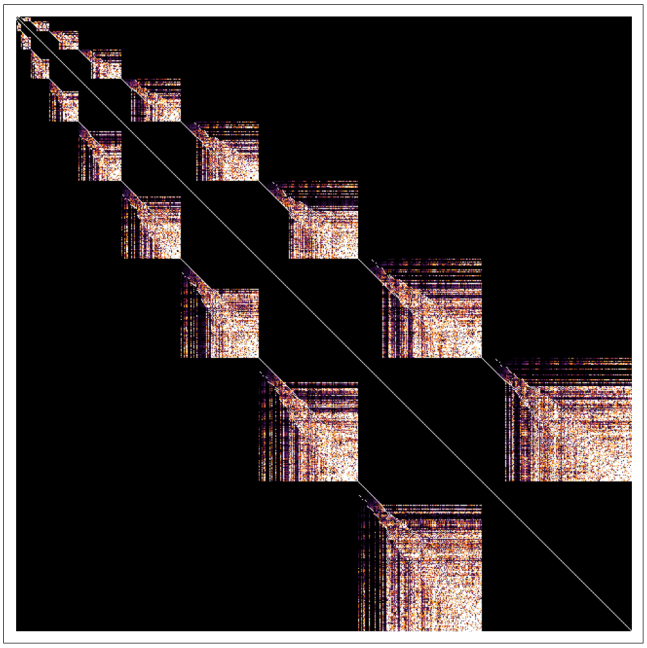
```
SetDirectory["~/Desktop/"];
```

- The following commands read in the kinetic energy and the interaction matrix elements from the two files and store them in two matrix-type variables TMat and VMat, respectively.

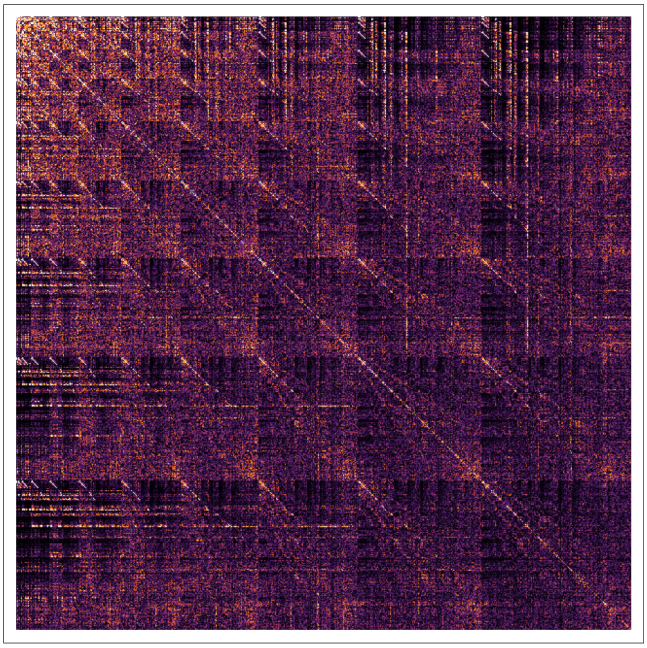
```
TMat = Import["kinetic_EMax20_hw20.dat", "Table"];
VMat = Import["chi2b3b_EMax20_hw20.dat", "Table"];
```

- First we visualize these matrices. We use the ArrayPlot[] command to plot the absolute values of the kinetic energy (TMat) and the potential matrix elements (VMat), respectively. What do you observe?

```
ArrayPlot[Abs[TMat], PlotRange → {0, 3}]
```



```
ArrayPlot[Abs[VMat], PlotRange → {0, 3}]
```



SRG Evolution

Formalism and Algorithm

- Now we implement the SRG-evolution of the Hamiltonian using the kinetic energy and potential matrix elements from above.
- We start from the evolution equation for the operators with a flow parameter a and expand the commutators

$$\frac{d}{da} H[a] = \\ = MN^2\,[[T, H[a]], H[a]] = \\ MN^2\,(T\,H[a]\times H[a] - 2\,H[a]\,T\,H[a] + H[a]\times H[a]\,T) \tag{1}$$

- By using the Jacobi HO basis representation this translates into a matrix evolution equation

$$\frac{d}{da} HMat[a] = \\ = MN^2\,(TMat.HMat[a].HMat[a] - \\ 2\,HMat[a].TMat.HMat[a] + HMat[a].HMat[a].TMat) \tag{2}$$

- We can solve this matrix evolution equation using the simplest possible algorithm for treating ordinary differential equations numerically, the so-called Euler algorithm. It is derived by simply replacing the derivative on the left-hand-side by a finite difference approximation for a small step aDel

$$\frac{d}{da}\,\text{HMat}[a] \longrightarrow \frac{1}{\text{aDel}}\,(\text{HMat}[a+\text{aDel}] - \text{HMat}[a]) \tag{3}$$

- After some simple algebra we obtain the following explicit equation, which gives the matrix HMat for flow parameter a+aDel based on the matrices TMat and HMat at flow parameter a, i.e., it brings us one step further in the flow parameter a.

$$\begin{aligned}&\text{HMat}[a + \text{aDel}] = \\ &\quad = \text{HMat}[a] + \text{aDel}\,\text{MN}^2\,(\text{TMat.HMat}[a].\text{HMat}[a] - \\ &\qquad 2\,\text{HMat}[a].\text{TMat.HMat}[a] + \text{HMat}[a].\text{HMat}[a].\text{TMat})\end{aligned} \tag{4}$$

Implementation

- We implement the Euler-type evolution directly. We set a small step size aDel and prepare the initial Hamiltonian matrix, given by the sum of the kinetic energy matrix and the interaction matrix.

```
a = 0.0;
aDel = 0.0001;
HMat = TMat + VMat;
```

- For storing the SRG-evolved interaction matrix elements, we use a set of variables VMatSRG[a], where a is the numerical value of the flow-parameter

```
Clear[VMatSRG];
VMatSRG[0.0] = VMat;
```

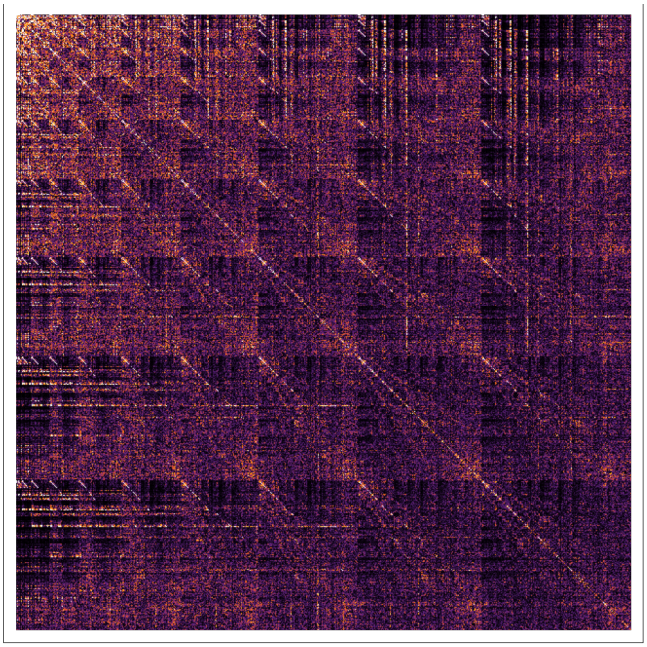
- We evaluate a single step of the Euler algorithm according to the above equation (4). Since the step size aDel has to be very small in order to make the algorithm work, we have to do many steps to evolve the matrices to sizable flow-parameters a. Therefore, we always do 500 evolution steps in a row using a simple Do-loop. The evaluation of this little program takes some time...

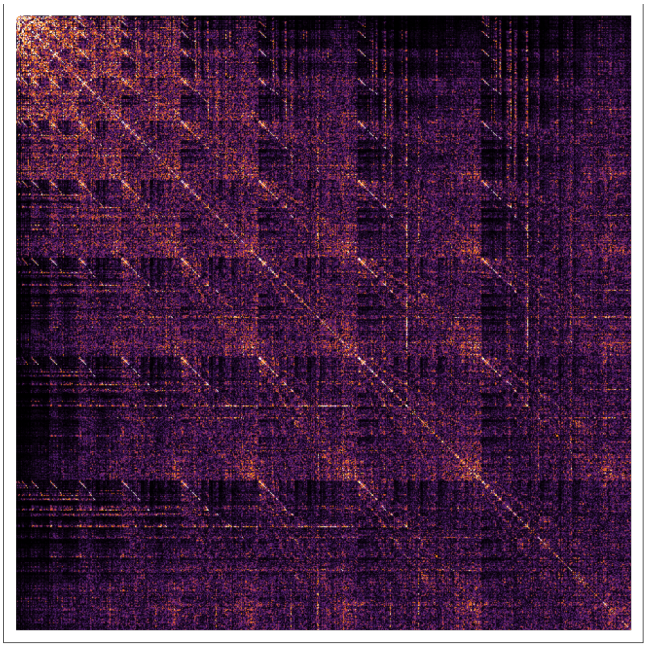
```
Do[
  HMatDel =
   aDel MN^2 ( TMat.HMat.HMat - 2 HMat.TMat.HMat + HMat.HMat.TMat );
  HMat = HMat + HMatDel;
  a = a + aDel;
  , {500}];

a = Round[a, 0.00001];
VMatSRG[a] = HMat - TMat;

Print["Evolution up to a=" <> ToString[a] <> " completed."];
```

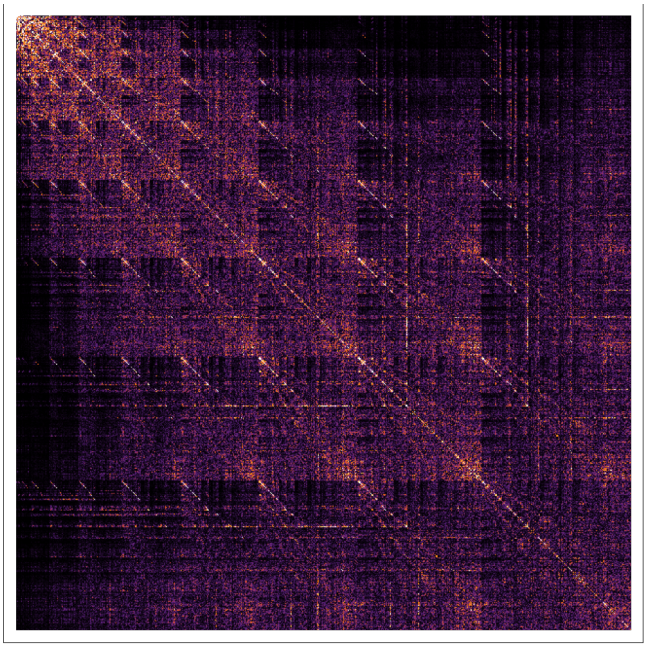
```
Evolution up to a=0.1 completed.
```

- Note that this code just continues the evolution from the latest values of the variables HMat and a. Thus, if you evaluate the cell a second or third time, you will continue the evolution to larger a. The intermediate results for the evolved interaction matrix elements are stored in the variables VMatSRG[0.05], VMatSRG[0.1],... Make sure that you evaluate the above cell at least twice!
- We can create plots of the initial interaction matrix elements and the evolved ones for direct comparison. What do you observe?

```
ArrayPlot[ Abs[VMatSRG[0.0]], PlotRange → {0, 3}]
ArrayPlot[ Abs[VMatSRG[0.05]], PlotRange → {0, 3}]
ArrayPlot[ Abs[VMatSRG[0.1]], PlotRange → {0, 3}]
```







No-Core Shell Model

- Finally, we can solve the exact Schrödinger equation for the triton via a matrix eigenvalue problem, which corresponds to a full-fledged No-Core Shell Model calculation.

Implementation

- In Mathematica this is essentially a one-line program once we have the Hamiltonian matrix elements available. We use the command Eigenvalues[] to obtain the 5 eigenvalues of lowest magnitude and select the smallest one from these, since we are only interested in the ground state.
- For the initial Hamiltonian we get

```
HMat = TMat + VMatSRG[0.0];
Min[Eigenvalues[HMat, -5]]
```

```
-8.07046
```

- For the SRG-evolved Hamiltonian we get for a=0.05

```
HMat = TMat + VMatSRG[0.05];
Min[Eigenvalues[HMat, -5]]
```

```
-8.08892
```

- For the SRG-evolved Hamiltonian we get for a=0.1

```
HMat = TMat + VMatSRG[0.1];
Min[Eigenvalues[HMat, -5]]
```

```
-8.08978
```

- The eigenvalues we obtain are almost the same for all three Hamiltonians, which is due to the unitarity of the transformation. Do you have any idea why they are not exactly the same?

Convergence

- We can also look at the convergence behavior of the different Hamiltonians. In order to do so, we extract from the full Hamiltonian matrix a sequence of submatrices that correspond to the different NMax-truncated model spaces for NMax=2,...,20. For this we need the sizes of the submatrices corresponding to the different NMax-truncated spaces, which are given in the following cell:

```
NMaxList = {2, 4, 6, 8, 10, 12, 14, 16, 18, 20};
DimList = {5, 15, 34, 64, 108, 169, 249, 351, 478, 632};
```

- Now we can extract a sequence of submatrices and solve the eigenvalue problem for each of them. This gives us the ground-state energies for the triton for a sequence of NMax values, which are returned by the following cell as a list with entries {NMax, EE0}. We start with the initial Hamiltonian:

```
EEList[0.0] = Table[
  HMat = Take[TMat + VMatSRG[0.0], DimList[[i]], DimList[[i]]];
  EE0 = Min[Eigenvalues[HMat, -5]];
  {NMaxList[[i]], EE0}, {i, Length[NMaxList]}]
```

```
{{2, 4.30575}, {4, 1.38829}, {6, -0.916138},
 {8, -2.29174}, {10, -4.03652}, {12, -5.2033}, {14, -6.36135},
 {16, -7.15481}, {18, -7.73525}, {20, -8.07046}}
```

- And now we do the same for the SRG-evolved Hamiltonians

```
EEList[0.05] = Table[
  HMat = Take[TMat + VMatSRG[0.05], DimList[[i]], DimList[[i]]];
  EE0 = Min[Eigenvalues[HMat, -5]];
  {NMaxList[[i]], EE0}, {i, Length[NMaxList]}]
```

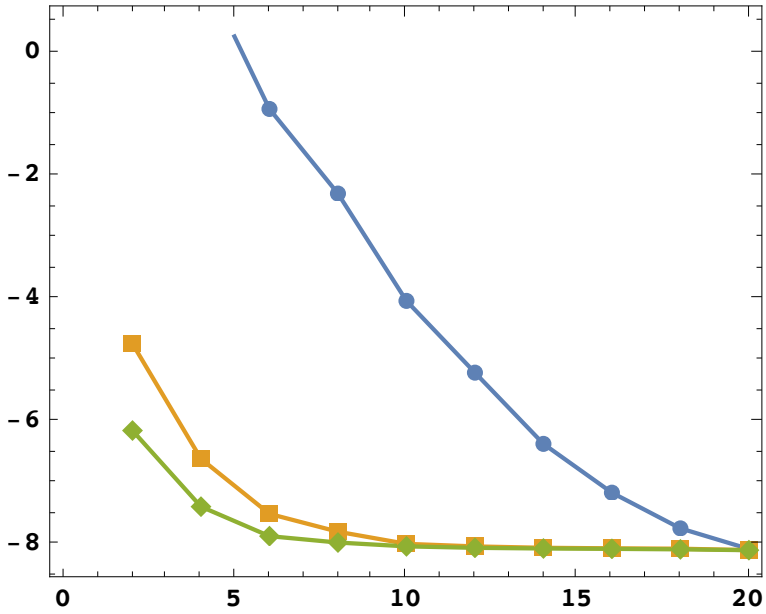
```
{{2, -4.73429}, {4, -6.60428}, {6, -7.49934},
 {8, -7.78952}, {10, -7.98678}, {12, -8.02854}, {14, -8.05087},
 {16, -8.05975}, {18, -8.06561}, {20, -8.08892}}
```

```
EEList[0.1] = Table[
  HMat = Take[TMat + VMatSRG[0.1], DimList[[i]], DimList[[i]]];
  EE0 = Min[Eigenvalues[HMat, -5]];
  {NMaxList[[i]], EE0}, {i, Length[NMaxList]}]
```

```
{{2, -6.14463}, {4, -7.38208}, {6, -7.86173},
 {8, -7.96549}, {10, -8.02765}, {12, -8.0515}, {14, -8.06224},
 {16, -8.06819}, {18, -8.07618}, {20, -8.08978}}
```

- We can visualize the results in a combined convergence plot for the three Hamiltonians, showing the ground-state energies as function of NMax. This is the type of plot that was shown in the lecture as well. What do you observe?

```
ListLinePlot[{EEList[0.0], EEList[0.05], EEList[0.1]}]
```



Next Steps

Fundamentals

- Implement the above SRG transformation and the NCSM solution in a programming language of your choice. You can copy/adapt this Mathematica implementation, but then make sure that you understand each and every line.

Algorithms

- Replace the simplistic Euler approach for solving the SRG initial-value problem with a more sophisticated ODE solver. How can you check the numerical accuracy of the numerical algorithm?
- Implement the Lanczos algorithm for solving the eigenvalue problem and explore its properties, particularly, the convergence of eigenvalues with the number of iterations and the possible loss of orthogonality due to round-off errors.

Physics

- Plot the ground state energy of the triton for different Nmax (e.g. Nmax=0,10,20) as function of the SRG flow parameter. Interpret your result.
- Play with the SRG evolution and study, e.g., what happens for larger flow parameters or what changes if you reverse the direction of the flow.
- Invent and test new generators for the SRG evolution.