Mathematica HEOM, general method with Matsubara frequencies and multiple baths per site

Written on 26.12.2014 by Tobias Kramer (email: tobias.kramer@mytum.de and tobias.kramer@jku.at), tidied up 2014-2023

This notebook implements the HEOM algorithm as described in
[1] Tanimura & Kubo “Time Evolution of a Quantum System in Contact with a Nearly Gaussian-Markoffian Noise Bath”, J. Phys. Soc. Jpn. 58, pp. 101-114 (1989)
[2] Tanimura “Stochastic Liouville, Langevin, Fokker–Planck, and Master Equation Approaches to Quantum Dissipative Systems”, J. Phys. Soc. Jpn. 75, 082001 (2006)
[3] Kreisbeck & Kramer “Long-Lived Electronic Coherence in Dissipative Exciton Dynamics of Light-Harvesting Complexes”, J. Phys. Chem. Lett.319 2828-2833 (2012)
[4] Kramer et al: “Efficient calculation of open quantum system dynamics and time-resolved spectroscopy with distributed memory HEOM (DM-HEOM)”
and further references therein.

This notebook is part of the Open Quantum Dynamics Lab, funded by the FWF through project P 35844 (project duration 2023-2026), project lead:
Univ.-Prof. Dr. Tobias Kramer
Dept. of Classical and Quantum Dynamics
Institute for Theoretical Physics
Johannes Kepler University Linz
Austria (contact: tobias.kramer@jku.at).
If you use this notebook for your research, please acknowledge this implementation and share your modifications publicly.
Citation and DOI numbers are provided by https://zenodo.org

Physical constants

```
In[1]:= ℏSI = 1.05457148`200*^-34;
        kBSI = 1.380650239684840458953781387048428305608616777679979316`200.*^-23;
        invcmtoJoule = 1.986456217327470903861498336971100097137908380913187891`200.*^-23;

In[4]:= PRECISION = $MachinePrecision;
```

Basic parameters of the system and baths

Hamiltonian (system part)

```
In[5]:= SITES = 2; (* Number of sites, including ground state and double exciton states *)
```

```
In[6]:= Hex = {{100, 50}, {50, 200}};
```

This is the complete Hamiltonian, including ground state and 2 exciton states

```
In[7]:= Hex // MatrixForm
Out[7]//MatrixForm=
```

$$\begin{pmatrix} 100 & 50 \\ 50 & 200 \end{pmatrix}$$

```
In[8]:= HMSI = SetPrecision[Hex * invcmtoJoule, PRECISION];
```

Integration time step

```
In[9]:= hSI = SetPrecision[5 * 10^-15, PRECISION]; (* time-step for integration [s] *)
```

Bath parameters

```
In[10]:= NMAX = 4; (* HEOM truncation depth *)
         MATSUBARAS = 1; (* Number of Matsubara frequencies *)
```

```
In[12]:= TSI = 200; (* temperature, common to all baths [K] *)
         νinv = 50 * 10^-15; (* inverse bath correlation time [s] *)
         λinvcm = 15; (* reorganization energy [1/cm] *)
         Ωinvcm = 0; (* shift of spectral density peak,
         use pairs of peaks +/- shifted for building structured J(ω) *)
```

```
In[16]:= βSI = SetPrecision[1 / (kBSI TSI), PRECISION];
         ν1SI = SetPrecision[2 π / (βSI ħSI), PRECISION];
         νSI = SetPrecision[1 / νinv, PRECISION];
         λSI = SetPrecision[λinvcm * invcmtoJoule, PRECISION];
         ΩSI = SetPrecision[Ωinvcm * invcmtoJoule / ħSI, PRECISION];
```

```
In[21]:= NUMBATHS = 2; (* total number of independent baths *)
         MAXBATHSPERSITE = 1 ;(* maximum number of baths coupling to a site *)
         BASTable = {
            {0}, (* C++ indexing starts at site 0 = ground state,
            -1 indicates NO coupling *)
            {1}};
         (* for compatibility construct also the reverse map: *)
         (* coupling: bath2sites⟦b⟧⟦s⟧ → bath b is coupled to site s *)
         bath2sites =
          DeleteCases[Table[Flatten[Drop[Position[BASTable, s], {}, -1]], {s, 0, SITES - 1}], {}]
Out[24]= {{1}, {2}}
```

```
In[25]:= bath2sites⟦1⟧ (* first bath connects to sites 1 in MM counting *)
         bath2sites⟦2⟧ (* second bath connects to sites 2 in MM counting) *)
Out[25]= {1}

Out[26]= {2}
```

Initialize each bath

No shift Ω for the moment...

```
In[27]:= νNUMBATHS[ν_, Ω_] = Table[ν, {i, 1, NUMBATHS}]
         λNUMBATHS[λ_] = Table[λ, {i, 1, NUMBATHS}]
Out[27]= {ν, ν}

Out[28]= {λ, λ}
```

HEOM: construction of auxiliary density operators (ADO) and indexing setup

```
In[29]:= ADOWIDTH = NUMBATHS * MATSUBARAS;
         ADOTuple =.;
         ADOTuple = {};
         For[nd = 0, nd ≤ NMAX,
           iplist = IntegerPartitions[nd, ADOWIDTH];
           For[i = 1, i ≤ Length[iplist],
            AppendTo[ADOTuple, Permutations[PadRight[iplist⟦i⟧, ADOWIDTH]]];
            i++];
           nd++];
         ADOTuple = Partition[Flatten[ADOTuple], ADOWIDTH];

In[34]:= ADOTupleDict = Table[FromCharacterCode[ADOTuple⟦i⟧], {i, 1, Length[ADOTuple]}];
         assoc = Association[Table[ADOTupleDict⟦i⟧ → i, {i, 1, Length[ADOTuple]}]];
```

To implement the PLUS/MINUS actions on index tuples, we define the DD possibilities to add/subtract 1 from the tuple elements.

```
In[36]:= null = Table[0, {i, 1, ADOWIDTH}];
         poperator = Permutations[Table[If[i == 1, 1, 0], {i, 1, ADOWIDTH}]];
         moperator = Permutations[Table[If[i == 1, -1, 0], {i, 1, ADOWIDTH}]];
```

For each tuple we now perform the PLUS operation and look up in the association map to which uid the PLUS operator leads.
Note that only index tuples with a depth < NMAX do have a PLUS link. Non-existing links are pointing to uid=-1.

```
In[39]:= PlusIndex =.;
         PlusIndex = Table[-1, {i, 1, ADOWIDTH * Length[ADOTuple]}];
         For[uid = 0, uid < Length[ADOTuple],
           For[j = 0, j < ADOWIDTH,
            tc = FromCharacterCode[ADOTuple[[uid + 1]] + poperator[[j + 1]]];
            PlusIndex[[ADOWIDTH * uid + j + 1]] = Lookup[assoc, tc, -1];
            j++];
           uid++];
         PlusIndex = Partition[PlusIndex, ADOWIDTH];
```

For each tuple we now perform the MINUS operation and look up in the association map to which uid the MINUS operator leads.
Note that only tuple elements >=1 do have a MINUS link. Non-existing links are pointing to uid=-1.

```
In[43]:= MinusIndex =.;
         MinusIndex = Table[-1, {i, 1, ADOWIDTH * Length[ADOTuple]}];
         For[uid = 0, uid < Length[ADOTuple],
           For[j = 0, j < ADOWIDTH,
            testADOTuple = ADOTuple[[uid + 1]] + moperator[[j + 1]];
            If[MemberQ[testADOTuple, -1],
             MinusIndex[[ADOWIDTH * uid + j + 1]] = -1;
             ,
             tc = FromCharacterCode[testADOTuple];
             MinusIndex[[ADOWIDTH * uid + j + 1]] = Lookup[assoc, tc, -1];
            ];
            j++];
           uid++];
         MinusIndex = Partition[MinusIndex, ADOWIDTH];
```

```
In[47]:= Print["Total number of ADOs: ", Length[ADOTupleDict]]

         Total number of ADOs: 15
```

Setup of the operators

```
In[48]:= νbk[b_, k_, ℏ_, ν_, λ_, β_, ν1_, Ω_] := If[k == 0, νNUMBATHS[ν, Ω][[b]], 2 π k / (β ℏ)]
```

```
In[49]:= cbk[b_, k_, ℏ_, ν_, λ_, β_, ν1_, Ω_] :=
          If[k == 0, νNUMBATHS[ν, Ω][[b]] × λNUMBATHS[λ][[b]] Cot[β ℏ νNUMBATHS[ν, Ω][[b]] / 2],
           4 λNUMBATHS[λ][[b]] × νNUMBATHS[ν, Ω][[b]]                  (2 π k / (β ℏ))
           ─────────────────────────────────────────  ──────────────────────────────────────────────── ]
                             β ℏ                       (2 π k / (β ℏ))^2 - (νNUMBATHS[ν, Ω][[b]])^2
```

```
In[50]:= bktt[b_, k_] := (b - 1) * MATSUBARAS + k + 1
```

```
In[51]:= kkOp[k_] := DiagonalMatrix[Table[If[i == k, 1, 0], {i, 1, SITES}]]
```

```
In[52]:= ϕk[k_, A_] := ⅈ (kkOp[k].A - A.kkOp[k])
         Vcross[k_, A_] := kkOp[k].A - A.kkOp[k]
         Vcirc[k_, A_] := kkOp[k].A + A.kkOp[k]
```

In[55]:=

$$\text{θkMA}[b\_,\ s\_,\ k\_,\ A\_,\ \hbar\_,\ \nu\_,\ \lambda\_,\ \beta\_,\ \nu 1\_,\ \Omega\_] := \text{If}\Big[k == 0,\ \left(\frac{i\,\lambda\,\nu\,\text{Cot}\left[\frac{\beta\,\nu\,\hbar}{2}\right]}{\hbar}\right)\text{Vcross}[\text{bath2sites}[[b]][[s]],\ A] + \frac{\text{Vcirc}[\text{bath2sites}[[b]][[s]],\ A]\ \nu\,\lambda}{\hbar},$$

$$\frac{4\,i\,\nu\,\lambda\ \nu\text{bk}[b,\ k,\ \hbar,\ \nu,\ \lambda,\ \beta,\ \nu 1,\ \Omega]}{\left(-\beta\,\nu^2\,\hbar^2 + \beta\,\hbar^2\,\nu\text{bk}[b,\ k,\ \hbar,\ \nu,\ \lambda,\ \beta,\ \nu 1,\ \Omega]^2\right)}\ \text{Vcross}[\text{bath2sites}[[b]][[s]],\ A]\Big]$$

The final HEOM equation (in similar notation to [R3])

In[56]:=

$$\text{dσdtMASC}[\text{HM}\_,\ \sigma\text{M}\_,\ \text{uid}\_,\ \hbar\_,\ \nu\_,\ \lambda\_,\ \beta\_,\ \nu 1\_,\ \Omega\_] := \Big(-\frac{i}{\hbar}\left(\text{HM}.\sigma\text{M}[[\text{uid}]] - \sigma\text{M}[[\text{uid}]].\text{HM}\right) -$$

$$\sigma\text{M}[[\text{uid}]] \times \sum_{b=1}^{\text{NUMBATHS}}\ \sum_{k=0}^{\text{MATSUBARAS}-1} \text{ADOTuple}[[\text{uid}]][[\text{bktt}[b,\ k]]] \times \nu\text{bk}[b,\ k,\ \hbar,\ \nu,\ \lambda,\ \beta,\ \nu 1,\ \Omega] -$$

$$\sum_{b=1}^{\text{NUMBATHS}}\ \sum_{s=1}^{\text{Length}[\text{bath2sites}[[b]]]} \left(\frac{2\ \lambda\text{NUMBATHS}[\lambda]\ [[b]]}{\beta\,\hbar^2\ \nu\text{NUMBATHS}[\nu,\ \Omega]\ [[b]]} - \sum_{k=0}^{\text{MATSUBARAS}-1} \frac{\text{cbk}[b,\ k,\ \hbar,\ \nu,\ \lambda,\ \beta,\ \nu 1,\ \Omega]}{\hbar\ \nu\text{bk}[b,\ k,\ \hbar,\ \nu,\ \lambda,\ \beta,\ \nu 1,\ \Omega]}\right)$$

$$\text{Vcross}[\text{bath2sites}[[b]][[s]],\ \text{Vcross}[\text{bath2sites}[[b]][[s]],\ \sigma\text{M}[[\text{uid}]]\ ]] +$$

$$\sum_{b=1}^{\text{NUMBATHS}}\ \sum_{s=1}^{\text{Length}[\text{bath2sites}[[b]]]}\ \sum_{k=0}^{\text{MATSUBARAS}-1} \text{Sqrt}[\ (\text{ADOTuple}[[\text{uid}]][[\text{bktt}[b,\ k]]] + 1)$$

$$\text{Abs}[\text{cbk}[b,\ k,\ \hbar,\ \nu,\ \lambda,\ \beta,\ \nu 1,\ \Omega]\ /\ \hbar]]$$

$$\phi\text{k}[\text{bath2sites}[[b]][[s]],\ \sigma\text{M}[[\ \text{PlusIndex}[[\text{uid}]][[\text{bktt}[b,\ k]]]\ ]]\ ]$$

$$+ \sum_{b=1}^{\text{NUMBATHS}}\ \sum_{s=1}^{\text{Length}[\text{bath2sites}[[b]]]}\ \sum_{k=0}^{\text{MATSUBARAS}-1} \text{Sqrt}[\ \text{ADOTuple}[[\text{uid}]][[\text{bktt}[b,\ k]]]\ /$$

$$\text{Abs}[\text{cbk}[b,\ k,\ \hbar,\ \nu,\ \lambda,\ \beta,\ \nu 1,\ \Omega]\ /\ \hbar]]\ \text{θkMA}[b,\ s,\ k,\ \sigma\text{M}[[\text{MinusIndex}[[\text{uid}]][[$$

$$\text{bktt}[b,\ k]]]]],\ \hbar,\ \nu\text{NUMBATHS}[\nu,\ \Omega]\ [[b]],\ \lambda\text{NUMBATHS}[\lambda]\ [[b]],\ \beta,\ \nu 1,\ \Omega]\Big)$$

Running the HEOM algorithm within *Mathematica*

Now use the matrices and include as last element a zero matrix, this ensures that whenever σM[-1] is called, it does not contribute, since the array index -1 maps to the last element (=the zeroed matrix).

In[57]:=
```
nM = DiagonalMatrix[Table[0, {i, 1, SITES}]]
σM0 = Table[nM, {i, 1, Length[ADOTuple]}];
```

Out[57]=
```
{{0, 0}, {0, 0}}
```

Set the initial density matrix

```
In[59]:= rho0 = DiagonalMatrix[Table[If[i == 1, 1, 0], {i, 1, SITES}]]
Out[59]= {{1, 0}, {0, 0}}

In[60]:= rho0 // MatrixForm
Out[60]//MatrixForm=
         ( 1  0 )
         ( 0  0 )
```

Initialize one member of the hierarchy with the initial density matrix, append the zero matrix at the end for the -1 links

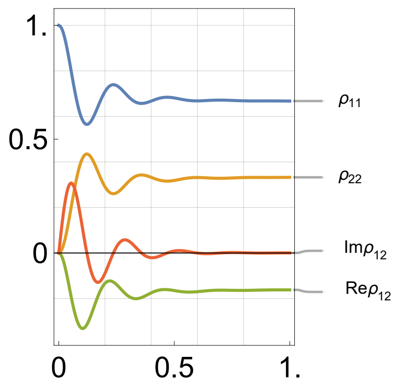
```
In[61]:= σM0[[1]] = rho0;
         AppendTo[σM0, nM];
```

Taylor integration

```
In[63]:= m = 200; (* number of steps *)
         YTA4 = T = Table[0, {m + 1}];
         T[[1]] = 0;
         YTA4[[1]] = σM0;
         Timing[
          For[j = 1, j ≤ m, j++,
           k1 = SetPrecision[hSI Table[dσdtMASC[HMSI, YTA4[[j]],       i, ħSI,
                νSI, λSI, βSI, ν1SI, ΩSI], {i, 1, Length[ADOTuple]}], PRECISION];
           AppendTo[k1, nM];
           YTA4[[j + 1]] = SetPrecision[YTA4[[j]] + k1 / 1 !, PRECISION];
           k2 = hSI Table[
              dσdtMASC[HMSI, k1, i, ħSI, νSI, λSI, βSI, ν1SI, ΩSI], {i, 1, Length[ADOTuple]}];
           AppendTo[k2, nM];
           YTA4[[j + 1]] = SetPrecision[YTA4[[j + 1]] + k2 / 2 !, PRECISION];
           k1 = hSI Table[
              dσdtMASC[HMSI, k2, i, ħSI, νSI, λSI, βSI, ν1SI, ΩSI], {i, 1, Length[ADOTuple]}];
           AppendTo[k1, nM];
           YTA4[[j + 1]] = SetPrecision[YTA4[[j + 1]] + k1 / 3 !, PRECISION];
           k2 = hSI Table[
              dσdtMASC[HMSI, k1, i, ħSI, νSI, λSI, βSI, ν1SI, ΩSI], {i, 1, Length[ADOTuple]}];
           AppendTo[k2, nM];
           YTA4[[j + 1]] = SetPrecision[YTA4[[j + 1]] + k2 / 4 !, PRECISION];
           T[[j + 1]] = hSI * j;
          ]]
Out[67]= {3.17, Null}
```

```
In[68]:= p = ListPlot[{
            Transpose[{T * 10^12, Re[YTA4[[All, 1, 1, 1]]]}],
            Transpose[{T * 10^12, Re[YTA4[[All, 1, 2, 2]]]}],
            Transpose[{T * 10^12, Re[YTA4[[All, 1, 1, 2]]]}],
            Transpose[{T * 10^12, Im[YTA4[[All, 1, 1, 2]]]}]
          }, PlotLabels → {Text["ρ11"], Text["ρ22"], Text["Reρ12"], Text["Imρ12"]},
          Joined → True, PlotRange → All, ImageSize → 220,
          Frame → True, AspectRatio → 1.4, LabelStyle → Directive[Black, 15],
          FrameTicks → {{{0, 0.5, 1.0}, None}, {{0, 0.5, 1.0}, None}}, GridLines → Automatic]
```

Out[68]=



```
In[69]:= (*Export["fig_oqsd.png",p,ImageResolution→200]*)
```