Simultaneous Fermion and Exciton Condensations from a Model Hamiltonian:

Published in *Physical Review B*:

DOI:

https://doi.org/10.1103/PhysRevB.00.005100

LeeAnn M. Sager and David A. Mazziotti

1 Input Values

```
> N,r:=4,8:
```

```
> spec_e,spec_l,spec_g,spec_G:=0,-0.5,-0.5,0.7:
```

2 Loading Necessary Functions

```
> mbasispsi := proc(n,r)
options ‘Copyright (c) David A. Mazziotti 2022‘;
local F, i, rseq, baseP, seqq, sett, k;
F := proc(a, b)
local an, bn;
an := nops(select(has, a, alpha));
bn := nops(select(has, b, alpha));
if an < bn then
  RETURN(false)
else
  RETURN(true)
end if;
end proc;
rseq := [seq(i, i = 1 .. r)];
baseP := combinat:-choose(rseq, n);
seqq := [seq(k = alpha[k], k = 1 .. r/2), seq(k = beta[k], k = r
    /2 + 1 .. r)];
```

```
baseP := subs(seqq, baseP);
baseP := sort(baseP, F);
seqq := [seq(alpha[k] = k, k = 1 .. r/2), seq(beta[k] = k, k = r
    /2 + 1 .. r)];
baseP := subs(seqq, baseP);
baseP := map(convert, baseP, set);
return baseP;
end:
```

```
> with(LinearAlgebra):
```

```
>
```

3 Defining all Possible States and Pairings

```
> list1:= mbasispsi(N,r):
```

```
> nops(list1);
```

$$70 \tag{1}$$

```
> down:=\{seq(i,i=1..N)\};
```

$$down := \{1, 2, 3, 4\} \tag{2}$$

```
> up:=\{seq(i,i=N+1..r)\}
```

$$up := \{5, 6, 7, 8\} \tag{3}$$

```
> BCS_pairs:=\{seq(\{2*i-1,2*i\},i=1..N)\};
```

$$BCS\_pairs := \{\{1, 2\}, \{3, 4\}, \{5, 6\}, \{7, 8\}\} \tag{4}$$

```
> L_pairs:=\{seq(\{i,i+N\},i=1..N)\};
```

$$L\_pairs := \{\{1, 5\}, \{2, 6\}, \{3, 7\}, \{4, 8\}\} \tag{5}$$

```
>
```

```
>
```

4 Defining the Basis States

4.1 Necessary Code

```
> num_BCS_pairs := proc(configuration)
options ‘Copyright (c) LeeAnn M. Sager and David A. Mazziotti
    2022‘;
global BCS_pairs;
local i,num_pairs;
num_pairs:=0;
for i from 1 to nops(BCS_pairs) do
   if nops(configuration intersect BCS_pairs[i])=2 then
      num_pairs := num_pairs + 1
   end if;
end do;
return num_pairs
end:
```

```
> num_BCS_pairs_down := proc(configuration)
options ‘Copyright (c) LeeAnn M. Sager and David A. Mazziotti
    2022‘;
global BCS_pairs;
local i,num_pairs,temp_set;
num_pairs:=0;
for i from 1 to nops(BCS_pairs) do
   temp_set:=configuration intersect BCS_pairs[i];
   if nops(temp_set)=2 then
      if evalb(temp_set[1] in down) and evalb(temp_set[2] in down)
           then
         num_pairs := num_pairs + 1
      end if;
   end if;
end do;
return num_pairs
end:
```

```
> Lipkin_like := proc(configuration)
options ‘Copyright (c) LeeAnn M. Sager and David A. Mazziotti
    2022‘;
global L_pairs;
local i;
for i from 1 to nops(L_pairs) do
   if nops(configuration intersect L_pairs[i])>1 then
      return false
   end if;
end do;
```

```
return true
end:

> num_up_paired := proc(configuration)
options ‘Copyright (c) LeeAnn M. Sager and David A. Mazziotti
    2022‘;
global L_pairs,N,r;
local i,final_num;
final_num:=0;
for i from 1 to N by 2 do
   if nops(configuration intersect \{i,i+N+1\})=2 then
     if not nops(configuration intersect \{i,i+1,i+N,i+N+1\})>=3
         then
        final_num:=final_num+1
     end if;
   end if;
end do;
for i from 2 to N by 2 do
   if nops(configuration intersect \{i,i+N-1\})=2 then
     if not nops(configuration intersect \{i-1,i,i+N-1,i+N\})>=3
         then
        final_num:=final_num+1
     end if;
   end if;
end do;
return final_num
end:

> num_paired_stacked := proc(configuration)
options ‘Copyright (c) LeeAnn M. Sager and David A. Mazziotti
    2022‘;
global L_pairs,N,r,BCS_pairs;
local i,final_num,stacked_set;
final_num:=0;
stacked_set:=\{\};
for i from 1 to nops(L_pairs) do
   if evalb(nops(configuration intersect L_pairs[i])=2) then
     stacked_set:=stacked_set union L_pairs[i];
   end if;
end do;
return nops(stacked_set)/2,num_BCS_pairs(stacked_set)/2
end:

> num_up := proc(configuration)
options ‘Copyright (c) LeeAnn M. Sager and David A. Mazziotti
    2022‘;
```

```
global up;
return nops(configuration intersect up)
end:
```

```
> get_matching_configs := proc(spec_up,spec_BCS_pairs,L_bool:=
    None,spec_num_paired_pairs:=None,spec_num_up_paired:=None)
options `Copyright (c) LeeAnn M. Sager and David A. Mazziotti
    2022`;
global BCS_pairs, L_pairs, up;
local i,out_list;
out_list:=\{\};
for i from 1 to nops(list1) do
   if num_up(list1[i])=spec_up then
      if num_BCS_pairs(list1[i])=spec_BCS_pairs then
         if spec_num_paired_pairs=None or num_paired_stacked(list1
             [i])[2]=spec_num_paired_pairs then
            if spec_num_up_paired=None or num_up_paired(list1[i])=
                spec_num_up_paired then
               if L_bool=None then
                  out_list := \{out_list[],list1[i]\};
               else
                  if L_bool then
                     if Lipkin_like(list1[i]) then
                        out_list := \{out_list[],list1[i]\};
                     end if;
                  else
                     if not Lipkin_like(list1[i]) then
                        out_list := \{out_list[],list1[i]\};
                     end if;
                  end if;
               end if;
            end if;
         end if;
      end if;
   end if;
end do;
return out_list
end:
```

```
> all_down:=get_matching_configs(0,2,None);
```

$$\mathit{all\_down} := \{\{1,2,3,4\}\} \tag{6}$$

```
> double_BCS:=get_matching_configs(2,2,false);
```

$$\mathit{double\_BCS} := \{\{1,2,5,6\},\{3,4,7,8\}\} \tag{7}$$

```
> double_L:=get_matching_configs(2,0,true);
```

$$\mathit{double\_L} := \{\{1,3,6,8\},\{1,4,6,7\},\{2,3,5,8\},\{2,4,5,7\}\} \tag{8}$$

```
> double_both:=get_matching_configs(2,2,true);
```

$$\mathit{double\_both} := \{\{1,2,7,8\},\{3,4,5,6\}\} \tag{9}$$

```
> all_up:=get_matching_configs(4,2,None);
```

$$\mathit{all\_up} := \{\{5,6,7,8\}\} \tag{10}$$

```
> basis:=[all_down,double_BCS,double_both,double_L,all_up];
```

$$\begin{aligned}&\mathit{basis} :=\\ &\quad[\{\{1,2,3,4\}\},\{\{1,2,5,6\},\{3,4,7,8\}\},\\ &\quad\{\{1,2,7,8\},\{3,4,5,6\}\},\\ &\quad\{\{1,3,6,8\},\{1,4,6,7\},\{2,3,5,8\},\{2,4,5,7\}\},\{\{5,6,7,8\}\}]\end{aligned} \tag{11}$$

```
>
```

5 Defining the Hamiltonian and Lowest Eigenvector

5.1 Necessary Code

```
> e_list:=[];
```

$$\mathit{e\_list} := [] \tag{12}$$

```
> for j from 1 to N do
   e_list := [e_list[],[-1,j,j]];
   e_list := [e_list[],[1,j+N,j+N]];
end do:
```

```
> nops(e_list);
```

$$8 \tag{13}$$

```
> l_list:=[];
```

$$l\_list := [] \tag{14}$$

```
> for p from 1 to N do
    for q from p+1 to N do
      l_list := [l_list[],[p,q,q+N,p+N]];
      l_list := [l_list[],[p+N,q+N,q,p]];
    end do;
  end do:

> nops(l_list);
```

$$12 \tag{15}$$

```
> g_list:=[];
```

$$g\_list := [] \tag{16}$$

```
> for p from 1 to N do
    for q from 1 to N do
      g_list := [g_list[],[p+N,q,q+N,p]];
      g_list := [g_list[],[p,q+N,q,p+N]];
    end do;
  end do:

> nops(g_list);
```

$$32 \tag{17}$$

```
> G_list:=[];
```

$$G\_list := [] \tag{18}$$

```
> for j from 1 to N do
    for k from 1 to N do
      G_list := [G_list[], [2*j-1,2*j,2*k,2*k-1]]
    end do;
  end do:

> nops(G_list);
```

$$16 \tag{19}$$

```
>
```

```
> get_Hamiltonian_value := proc(left_val,right_val)
options ‘Copyright (c) LeeAnn M. Sager and David A. Mazziotti
    2022‘;
global e,l,G,g,l_list,G_list,g_list,N,r;
local i,i1,j,k,e_value,l_value,G_value,g_value,temp_ops,
    temp_ops_e,temp_ops_G,temp_ops_g,rv,temp_list_l,temp_list_G,
    temp_list_g,temp_list_e;
temp_list_e := [];
temp_list_l := [];
temp_list_G := [];
temp_list_g := [];
e_value := 0;
l_value := 0;
g_value := 0;
G_value := 0;
for i1 from 1 to nops(e_list) do
   temp_ops_e := e_list[i1];
   if evalb(temp_ops_e[3] in right_val) then
      rv := right_val minus \{temp_ops_e[3]\} union \{temp_ops_e
          [2]\};
      if rv = left_val then
         e_value := e_value + temp_ops_e[1];
         temp_list_e := [temp_list_e[],temp_ops_e];
      end if;
   end if;
end do;
for i from 1 to nops(l_list) do
   temp_ops := l_list[i];
   if evalb(temp_ops[3] in right_val) and evalb(temp_ops[4] in
       right_val) then
      rv := right_val minus \{temp_ops[4]\} minus \{temp_ops[3]\}
          union \{temp_ops[2]\} union \{temp_ops[1]\};
      if rv = left_val then
         l_value := l_value+1;
         temp_list_l := [temp_list_l[],temp_ops];
      end if;
   end if;
end do;
for j from 1 to nops(G_list) do
   temp_ops_G := G_list[j];
   if evalb(temp_ops_G[3] in right_val) and evalb(temp_ops_G[4] in
        right_val) then
      rv := right_val minus \{temp_ops_G[4]\} minus \{temp_ops_G
          [3]\} union \{temp_ops_G[2]\} union \{temp_ops_G[1]\};
      if rv = left_val then
         G_value := G_value+1;
         temp_list_G := [temp_list_G[],temp_ops_G];
      end if;
   end if;
end do;
for k from 1 to nops(g_list) do
   temp_ops_g := g_list[k];
   if evalb(temp_ops_g[3] in right_val) and evalb(temp_ops_g[4] in
        right_val) then
      rv := right_val minus \{temp_ops_g[4]\} minus \{temp_ops_g
          [3]\} union \{temp_ops_g[2]\} union \{temp_ops_g[1]\};
      if rv = left_val then
         g_value := g_value+1;
         temp_list_g := [temp_list_g[],temp_ops_g];
      end if;
   end if;
end do;
return e_value/2*e+l_value*l-G_value*G+g_value*g/2
end:

> get_cont_Hamiltonian_value:=proc(left_basis,right_basis)
options ‘Copyright (c) LeeAnn M. Sager and David A. Mazziotti
    2022‘;
global e,l,G,l_list,G_list,N,list1;
local i,j,local_sum;
local_sum:=0;
for i from 1 to nops(left_basis) do
   for j from 1 to nops(right_basis) do
      local_sum:=local_sum+get_Hamiltonian_value(left_basis[i],
          right_basis[j]);
   end do:
end do:
return local_sum/(sqrt(nops(left_basis))*sqrt(nops(right_basis)))
end:

> get_Ham_small := proc(basis)
options ‘Copyright (c) LeeAnn M. Sager and David A. Mazziotti
    2022‘;
global e,l,G,l_list,G_list,N,list1;
local out_matrix,i,j,left_value,right_value;
out_matrix := Matrix(nops(basis));
for i from 1 to nops(basis) do
   for j from i to nops(basis) do
      left_value:=basis[i];
      right_value:=basis[j];
      out_matrix[i,j]:=get_cont_Hamiltonian_value(left_value,
          right_value);
```

```
      out_matrix[j,i]:=out_matrix[i,j];
   end do;
end do;
return out_matrix
end:
```

```
> get_min_eig_vec := proc(matrix,spec_e:=None,spec_l:=None,spec_g
    :=None,spec_G:=None)
options ‘Copyright (c) LeeAnn M. Sager and David A. Mazziotti
    2022‘;
global e,l,G,g;
local matrix_1,stuff,index,eval,evec;
if evalb(spec_e=None) then
   matrix_1:=matrix;
else:
   matrix_1:=evalf(subs(e=spec_e,l=spec_l,g=spec_g,G=spec_G,matrix
       ));
end if;
stuff := Eigenvectors(Matrix(matrix_1, shape=symmetric));
index := ListTools[Search](min(Re(stuff[1])), convert(Re(stuff
    [1]),list));
eval := stuff[1][index];
evec := stuff[2][..,index];
return evalf(eval),evalf(evec)
end:
```

```
> small_ham:=get_Ham_small(basis);
```

$$small\_ham := \begin{bmatrix} -2e-2G & -G\sqrt{2} & \frac{(2l-2G)\sqrt{2}}{2} & 2l & 0 \\ -G\sqrt{2} & -2G+2g & -2G & 0 & -G\sqrt{2} \\ \frac{(2l-2G)\sqrt{2}}{2} & -2G & -2G & 2g\sqrt{2} & \frac{(2l-2G)\sqrt{2}}{2} \\ 2l & 0 & 2g\sqrt{2} & 2g & 2l \\ 0 & -G\sqrt{2} & \frac{(2l-2G)\sqrt{2}}{2} & 2l & 2e-2G \end{bmatrix} \tag{20}$$

```
> mat:=evalf(subs(e=spec_e,l=spec_l,g=spec_g,G=spec_G,small_ham))
    ;
```

$$mat := \tag{21}$$

$$\begin{bmatrix} -1.4 & -0.9899494934 & -1.697056274 & -1.0 & 0.0 \\ -0.9899494934 & -2.4 & -1.4 & 0.0 & -0.9899494934 \\ -1.697056274 & -1.4 & -1.4 & -1.414213562 & -1.697056274 \\ -1.0 & 0.0 & -1.414213562 & -1.0 & -1.0 \\ 0.0 & -0.9899494934 & -1.697056274 & -1.0 & -1.4 \end{bmatrix}$$

```
> val,vec:=get_min_eig_vec(mat);
```

$$val, vec := -5.79093396567645, \begin{bmatrix} 0.406197457757534 \\ 0.474341254242352 \\ 0.574449953769344 \\ 0.339138431559772 \\ 0.406197457757535 \end{bmatrix} \tag{22}$$

6 Get the D2 Matrix and Maximum Eigenvalue

6.1 Necessary Code

```
> get_D2_value:=proc(cre_cre,anh_anh,basis,vector)
options ‘Copyright (c) LeeAnn M. Sager and David A. Mazziotti
    2022‘;
global e,l,G,l_list,G_list,N,list1;
local i,j,i1,j1,sum_value,rv,lv,rv_value,lv_value;
sum_value:=0;
for i from 1 to nops(basis) do
   for j from 1 to nops(basis[i]) do
      for i1 from 1 to nops(basis) do
         for j1 from 1 to nops(basis[i1]) do
            if evalb(anh_anh[1] in basis[i][j]) and evalb(anh_anh
                [2] in basis[i][j]) then
               rv:= basis[i][j] minus \{anh_anh[1]\} minus \{
                   anh_anh[2]\} union \{cre_cre[1]\} union \{
                   cre_cre[2]\};
               lv:=basis[i1][j1];
               rv_value:=evalf(vector[i]/sqrt(nops(basis[i])));
               lv_value:=evalf(vector[i1]/sqrt(nops(basis[i1])));
               if evalb(rv=lv) then
                  sum_value:=sum_value+rv_value*lv_value;
               end if;
            end if;
         end do;
      end do;
   end do;
end do:
return sum_value
end:

> get_D2 := proc(basis,vector)
options ‘Copyright (c) LeeAnn M. Sager and David A. Mazziotti
    2022‘;
```

```
global e,l,G,l_list,G_list,N,list1;
local out_matrix,i,j,left_value,right_value;
out_matrix := Matrix(nops(BCS_pairs));
for i from 1 to nops(BCS_pairs) do
   for j from i to nops(BCS_pairs) do
      left_value:=BCS_pairs[i];
      right_value:=BCS_pairs[j];
      out_matrix[i,j]:=get_D2_value(left_value,right_value,basis,
          vector);
      out_matrix[j,i]:=out_matrix[i,j];
   end do;
end do;
return out_matrix
end:
```

```
> get_max_eig_vec := proc(matrix,spec_e:=None,spec_l:=None,spec_g
    :=None,spec_G:=None)
options ‘Copyright (c) LeeAnn M. Sager and David A. Mazziotti
    2022‘;
global e,l,G,g;
local matrix_1,stuff,index,eval,evec;
if evalb(spec_e=None) then
   matrix_1:=matrix;
else
   matrix_1:=subs(e=spec_e,l=spec_l,g=spec_g,G=spec_G,matrix);
end if;
stuff := Eigenvectors(Matrix(matrix_1, shape=symmetric));
index := ListTools[Search](max(Re(stuff[1])), convert(Re(stuff
    [1]),list));
eval := stuff[1][index];
evec := stuff[2][..,index];
return evalf(eval),evalf(evec)
end:
```

```
>
```

```
> D2:=get_D2(basis,vec);
```

$$D2 := \tag{23}$$

$$\begin{bmatrix} 0.442492561973171 & 0.272485311426638 & 0.329992749294427 & 0.272485311495126 \\ 0.272485311426638 & 0.442492561973171 & 0.272485311495126 & 0.329992749294427 \\ 0.329992749294427 & 0.272485311495126 & 0.442492561973172 & 0.272485311426638 \\ 0.272485311495126 & 0.329992749294427 & 0.272485311426638 & 0.442492561973172 \end{bmatrix}$$

```
> analD:=get_max_eig_vec(D2)[1];
```

$$analD := 1.31745593418936 \tag{24}$$

7 Get the Modified G2 Matrix and Maximum Eigenvalue

7.1 Necessary Code

```
> get_D1_value:=proc(cre,anh,basis,vector)
options `Copyright (c) LeeAnn M. Sager and David A. Mazziotti
    2022`;
global e,l,G,l_list,G_list,N,list1;
local i,j,i1,j1,sum_value,rv,lv,rv_value,lv_value;
sum_value:=0;
for i from 1 to nops(basis) do
   for j from 1 to nops(basis[i]) do
      for i1 from 1 to nops(basis) do
         for j1 from 1 to nops(basis[i1]) do
            if evalb(anh[1] in basis[i][j]) then
               rv:= basis[i][j] minus \{anh[1]\} union \{cre[1]\};
               lv:=basis[i1][j1];
               rv_value:=evalf(vector[i]/sqrt(nops(basis[i])));
               lv_value:=evalf(vector[i1]/sqrt(nops(basis[i1])));
               if evalb(rv=lv) then
                  sum_value:=sum_value+rv_value*lv_value;
               end if;
            end if;
         end do;
      end do;
   end do;
end do:
return sum_value
end:

> get_G2_block:=proc(p,q,basis,vector)
options `Copyright (c) LeeAnn M. Sager and David A. Mazziotti
    2022`;
global e,l,G,g,N,r;
local out_mat;
out_mat:=Matrix(4);
if p=q then
   out_mat[1,1]:=get_D1_value(\{p\},\{p\},basis,vector);
   out_mat[1,3]:=get_D1_value(\{p\},\{p+N\},basis,vector);
   out_mat[1,4]:=get_D2_value(\{p,p+N\},\{p+N,p\},basis,vector);
   out_mat[2,2]:=get_D1_value(\{p\},\{p\},basis,vector)-
       get_D2_value(\{p,p+N\},\{p,p+N\},basis,vector);
   out_mat[2,4]:=get_D1_value(\{p\},\{p+N\},basis,vector);
   out_mat[3,1]:=get_D1_value(\{p+N\},\{p\},basis,vector);
```

```
  out_mat[3,3]:=get_D1_value(\{p+N\},\{p+N\},basis,vector)-
     get_D2_value(\{p+N,p\},\{p+N,p\},basis,vector);
  out_mat[4,1]:=get_D2_value(\{p+N,p\},\{p,p+N\},basis,vector);
  out_mat[4,2]:=get_D1_value(\{p+N\},\{p\},basis,vector);
  out_mat[4,4]:=get_D1_value(\{p+N\},\{p+N\},basis,vector);
else:
  out_mat[1,1]:=get_D2_value(\{p,q\},\{q,p\},basis,vector);
  out_mat[1,2]:=get_D2_value(\{p,q+N\},\{q,p\},basis,vector);
  out_mat[1,3]:=get_D2_value(\{p,q\},\{q+N,p\},basis,vector);
  out_mat[1,4]:=get_D2_value(\{p,q+N\},\{q+N,p\},basis,vector);
  out_mat[2,1]:=get_D2_value(\{p,q\},\{q,p+N\},basis,vector);
  out_mat[2,2]:=get_D2_value(\{p,q+N\},\{q,p+N\},basis,vector);
  out_mat[2,3]:=get_D2_value(\{p,q\},\{q+N,p+N\},basis,vector);
  out_mat[2,4]:=get_D2_value(\{p,q+N\},\{q+N,p+N\},basis,vector);
  out_mat[3,1]:=get_D2_value(\{p+N,q\},\{q,p\},basis,vector);
  out_mat[3,2]:=get_D2_value(\{p+N,q+N\},\{q,p\},basis,vector);
  out_mat[3,3]:=get_D2_value(\{p+N,q\},\{q+N,p\},basis,vector);
  out_mat[3,4]:=get_D2_value(\{p+N,q+N\},\{q+N,p\},basis,vector);
  out_mat[4,1]:=get_D2_value(\{p+N,q\},\{q,p+N\},basis,vector);
  out_mat[4,2]:=get_D2_value(\{p+N,q+N\},\{q,p+N\},basis,vector);
  out_mat[4,3]:=get_D2_value(\{p+N,q\},\{q+N,p+N\},basis,vector);
  out_mat[4,4]:=get_D2_value(\{p+N,q+N\},\{q+N,p+N\},basis,vector
     );
end if;
return out_mat
end:

> get_G2 := proc(basis,vector)
options ‘Copyright (c) LeeAnn M. Sager and David A. Mazziotti
   2022‘;
global e,l,G,l_list,G_list,N,list1;
local out_matrix,i,j,left_value,right_value;
out_matrix := Matrix(4*N);
for i from 1 to N do
  out_matrix[4*(i-1)+1..4*i,4*(i-1)+1..4*i]:=get_G2_block(i,i,
     basis,vector);
  for j from i+1 to N do
    out_matrix[4*(i-1)+1..4*i,4*(j-1)+1..4*j]:=get_G2_block(i,j,
       basis,vector);
    out_matrix[4*(j-1)+1..4*j,4*(i-1)+1..4*i]:=out_matrix[4*(i
       -1)+1..4*i,4*(j-1)+1..4*j];
  end do;
end do;
return out_matrix
end:
```

```
> get_G2_mod:=proc(p,q,basis,vector)
options `Copyright (c) LeeAnn M. Sager and David A. Mazziotti
    2022`;
global e,l,G,g,N,r;
local out_mat;
out_mat:=Matrix(4);
out_mat[1,1]:=get_D1_value(\{p\},\{p\},basis,vector)*get_D1_value
    (\{q\},\{q\},basis,vector);
out_mat[1,2]:=get_D1_value(\{p\},\{p\},basis,vector)*get_D1_value
    (\{q\},\{q+N\},basis,vector);
out_mat[1,3]:=get_D1_value(\{p\},\{p\},basis,vector)*get_D1_value
    (\{q+N\},\{q\},basis,vector);
out_mat[1,4]:=get_D1_value(\{p\},\{p\},basis,vector)*get_D1_value
    (\{q+N\},\{q+N\},basis,vector);
out_mat[2,1]:=get_D1_value(\{p\},\{p+N\},basis,vector)*
    get_D1_value(\{q\},\{q\},basis,vector);
out_mat[2,2]:=get_D1_value(\{p\},\{p+N\},basis,vector)*
    get_D1_value(\{q\},\{q+N\},basis,vector);
out_mat[2,3]:=get_D1_value(\{p\},\{p+N\},basis,vector)*
    get_D1_value(\{q+N\},\{q\},basis,vector);
out_mat[2,4]:=get_D1_value(\{p\},\{p+N\},basis,vector)*
    get_D1_value(\{q+N\},\{q+N\},basis,vector);
out_mat[3,1]:=get_D1_value(\{p+N\},\{p\},basis,vector)*
    get_D1_value(\{q\},\{q\},basis,vector);
out_mat[3,2]:=get_D1_value(\{p+N\},\{p\},basis,vector)*
    get_D1_value(\{q\},\{q+N\},basis,vector);
out_mat[3,3]:=get_D1_value(\{p+N\},\{p\},basis,vector)*
    get_D1_value(\{q+N\},\{q\},basis,vector);
out_mat[3,4]:=get_D1_value(\{p+N\},\{p\},basis,vector)*
    get_D1_value(\{q+N\},\{q+N\},basis,vector);
out_mat[4,1]:=get_D1_value(\{p+N\},\{p+N\},basis,vector)*
    get_D1_value(\{q\},\{q\},basis,vector);
out_mat[4,2]:=get_D1_value(\{p+N\},\{p+N\},basis,vector)*
    get_D1_value(\{q\},\{q+N\},basis,vector);
out_mat[4,3]:=get_D1_value(\{p+N\},\{p+N\},basis,vector)*
    get_D1_value(\{q+N\},\{q\},basis,vector);
out_mat[4,4]:=get_D1_value(\{p+N\},\{p+N\},basis,vector)*
    get_D1_value(\{q+N\},\{q+N\},basis,vector);
return out_mat
end:

> get_mod_G2 := proc(basis,vector)
options `Copyright (c) LeeAnn M. Sager and David A. Mazziotti
    2022`;
global e,l,G,l_list,G_list,N,list1;
local out_matrix,i,j,left_value,right_value;
```

```
out_matrix := Matrix(4*N);
for i from 1 to N do
   out_matrix[4*(i-1)+1..4*i,4*(i-1)+1..4*i]:=get_G2_block(i,i,
       basis,vector)-get_G2_mod(i,i,basis,vector);
   for j from i+1 to N do
      out_matrix[4*(i-1)+1..4*i,4*(j-1)+1..4*j]:=get_G2_block(i,j,
          basis,vector)-get_G2_mod(i,j,basis,vector);
      out_matrix[4*(j-1)+1..4*j,4*(i-1)+1..4*i]:=out_matrix[4*(i
          -1)+1..4*i,4*(j-1)+1..4*j];
   end do;
end do;
return out_matrix
end:

>

> mod_G2:=get_mod_G2(basis,vec):

> analG:=get_max_eig_vec(mod_G2)[1];
```

$$analG := 1.32602904919025 \tag{25}$$

```
>
```