Orthogonalisation m=0

```
l = 3;
m = 1;
Nb = 100;
(**)
p = 1;
h = 1;
```

In[•]:=
```
I1[p1_, p2_] := Beta[p1 + 1, p2 - p1 - 1] * 2 ^ (p1 - p2 + 1);
I2[p1_, p2_] := I1[p1, p2 + 1];
(**)
C0[n_] := 8 * n ^ 2;
C1[n_] := 2 (1 + 2 * n * (n + 1) - 3 (2 n + 1));
C2[n_] := 2 (1 - 2 (n + 1));
(**)
K[k_, n_] :=
  -Which[n == 0 && m == 0, C1[n] * I1[k + n, k + n + 2 p + 2] + C2[n] * I1[k + n + 1, k + n + 2 p + 2] -
     l (l + 1) I1[k + n, k + n + 2 p], n > 0 && m == 0, C0[n] * I1[k + n - 1, k + n + 2 p + 2] +
     C1[n] * I1[k + n, k + n + 2 p + 2] + C2[n] * I1[k + n + 1, k + n + 2 p + 2] -
     l (l + 1) I1[k + n, k + n + 2 p], n > 0 && m > 0, C0[n] * I1[k + n - 1, k + n + 2 p + 2] +
     C1[n] * I1[k + n, k + n + 2 p + 2] + C2[n] * I1[k + n + 1, k + n + 2 p + 2] -
     l (l + 1) I1[k + n, k + n + 2 p] - m ^ 2 * I2[k + n - 1, k + n + 2 p]];
DotC[k_, n_] := Which[k == 0 && n == 0, K[0, 0] + p K[0, 1] + p K[1, 0] + p ^ 2 K[1, 1],
   k == 0 && n > 0, K[0, n + 1] + p K[1, n + 1], k > 0 && n == 0,
   K[k + 1, 0] + p K[k + 1, 1], k > 0 && n > 0, K[k + 1, n + 1]];
```

In[•]:=
```
(*https://reference.wolfram.com/language/ParallelTools/tutorial/
  ParallelEvaluation.html*)
n0 = If[m == 0, 0, 1];
MatC = ParallelTable[DotC[k, n], {k, n0, Nb}, {n, n0, Nb}];
(**)
invdet = (1 / Det[MatC]) // N
(*https://mathematica.stackexchange.com/questions/52367/how-
  to-estimate-the-matrix-condition-number-in-the-2-norm*)
(*k2=First@#/Last@#&@SingularValueList[MatC]//N*)
```

Out[•]=

$1.430228866844276 \times 10^{5930}$

In[•]:=
```
R1 = CholeskyDecomposition[MatC];
Y1 = Inverse[R1];
```

Potential basis

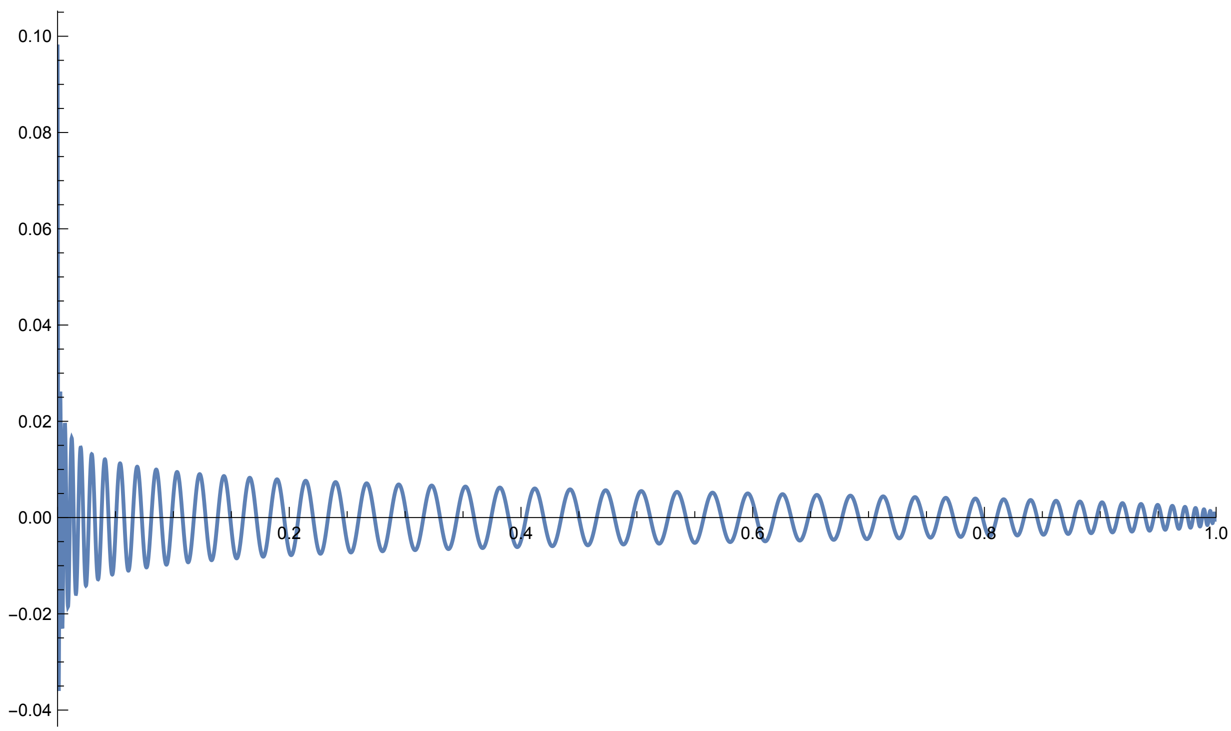
Orthonormal basis

In[∘]:=
```
fbare[x_, n_] := If[n > 0, (x) ^n, 1];
ft[x_, n_] := If[n == 0, fbare[x, 0] + p fbare[x, 1], fbare[x, n + 1]];
(**)
fGS[x_, n_] := Module[{y, z}, Clear[y];
   Clear[z];
   y = ParallelSum[Y1[[k + 1 - n0, n + 1 - n0]] * ft[x, k], {k, n0, n}];
   z = y // Simplify;
   Return[z]];
```

In[∘]:=
```
ξ[x_] := (1 + h x) / (1 - x);
Nr = Nb;
fg = fGS[x, Nr] / (ξ[x] + h);
nbx = Max[Nr * 100, 100];
tab = ParallelTable[{i / nbx, If[i < nbx, fg /. x → i / nbx, 0]}, {i, 0, nbx}] // N;
```

In[∘]:=
```
ListPlot[tab, Joined → True, ImageSize → Full,
 PlotRange → {{0, 1}, All}, ImageSize → Full]
```

Out[∘]=



Interpolate basis

In[ ]:=
```
TabSingle[Nr_] := Module[{fg, nbx, tab}, Clear[fg];
    Clear[nbx];
    fg = fGS[x, Nr] / (ξ[x] + h);
    nbx = Max[Nr * 100, 100];
    tab = ParallelTable[{i / nbx, If[i < nbx, fg /. x → i / nbx, 0]}, {i, 0, nbx}] // N;
    Return[tab]];
(**)
TabAll[Nmax_] := Table[TabSingle[Nr], {Nr, n0, Nmax}];
```

Store interpolated basis

In[ ]:=
```
nmax = Nb;
tabAll = TabAll[nmax];
```

In[ ]:=
```
(*https://sites.psu.edu/charlesammon/2017/07/05/writing-
  data-to-simple-hdf5-files-with-mathematica/*)
Do[Export[NotebookDirectory[] <> "h_1/l_" <> ToString[l] <> "/m_" <> ToString[m] <>
    "/F_l_" <> ToString[l] <> "_m_" <> ToString[m] <> "_n_" <> ToString[n] <> ".hdf5",
  tabAll⟦n + 1 - n0⟧, {"Datasets", "InterpolationTable"}], {n, n0, nmax}]
```