In[30]:=
```
SetOptions[SelectedNotebook[],
 PrintingStyleEnvironment → "Printout", ShowSyntaxStyles → True]
```

In[31]:=
```
<< Notation`
```

In[32]:=
```
(* Simplify the notation for the
 formulas: index vectors based on a subscript notation *)
Notation[ x_n_ ⟺ x_⟦n_⟧ ]
```

In[33]:=
```
α = Reverse@{13.00773, 1.962079, 0.444529, 0.1219492} & /@ Range[1, 2] // Flatten
```

Out[33]=
```
{0.121949, 0.444529, 1.96208, 13.0077, 0.121949, 0.444529, 1.96208, 13.0077}
```

In[34]:=
```
MatrixNormalize[v_, M_] := v / Sqrt[v.M.v];
```

In[35]:=
```
SCFLoop[d_, ϵ_, α_] := Module[{Z, K, R, S, T, V, H, Q, P, G, F, λ, ψ, ξ},
  P = {{0, 0, 0}, {d, 0, 0}}; (* Atomic positions *)
  Z = Partition[Table[#, {n, 1, 4}] & /@ P // Flatten, 3];

  (* Array of positions for the coefficients α *)
  K = Array[Exp[-α#1 α#2 / (α#1 + α#2) Norm[Z#1 - Z#2]^2] &, {8, 8}];
  (* Coefficients matrix K_{p,q} *)
  R = Array[(α#1 Z#1 + α#2 Z#2) / (α#1 + α#2) &, {8, 8}]; (* Distances matrix R_{p,q} *)
  S = Array[(π / (α#1 + α#2))^(3/2) K#1,#2 &, {8, 8}];
  (* Overlap matrix S_{p,q} *)

  T = Array[α#1 α#2 / (α#1 + α#2) (3 - 2 α#1 α#2 / (α#1 + α#2) Norm[Z#1 - Z#2]^2) S#1,#2 &, {8, 8}];
  (* Kinetic energy matrix T_{p,q} *)
  V = Total@Table[Array[If[Norm[R#1,#2 - U] == 0, -2 π / (α#1 + α#2) K#1,#2,
        -S#1,#2 1 / Norm[R#1,#2 - U] Erf[Sqrt[α#1 + α#2] Norm[R#1,#2 - U]]] &, {8, 8}], {U, P}];
  (* Potential energy matrix V_{p,q} *)
  H = T + V; (* Hamiltonian matrix H_{p,q} *)

  Q = Array[S#1,#3 S#2,#4 If[Norm[R#1,#3 - R#2,#4] == 0, 2 / Sqrt[π] Sqrt[(α#1 + α#3) (α#2 + α#4) / (α#1 + α#3 + α#2 + α#4)],
      1 / Norm[R#1,#3 - R#2,#4] Erf[Sqrt[(α#1 + α#3) (α#2 + α#4) / (α#1 + α#3 + α#2 + α#4)] Norm[R#1,#3 - R#2,#4]]] &,
    {8, 8, 8, 8}]; (* Q_{p,r,q,s} matrix, it's an 8 by 8
   matrix containing 8 by 8 matrices *)
```

```
P = ConstantArray[0., {8, 8}];
(* Initial guess for the density matrix *)
(* Perform a reset before the while-loop to assert correctness *)
λ = ConstantArray[0., {8}];
ξ = λ + ϵ;

(* While the new and previous eigenvalues are still different enough... *)
While[Norm[ξ - λ] > ϵ,
 ξ = λ; (* Overwrite the previous eigenvalues *)
 G = Array[∑_{r=1}^{8} ∑_{s=1}^{8} P_{r,s} (2 Q_{#1,r,#2,s} - Q_{#1,r,s,#2}) &, {8, 8}] (* G_{p,q} matrix *);
 F = H + 1/2 G; (* Compute the Fock operator from the G matrix *)
 {λ, ψ} = Eigensystem[{F, S}];
 (* Calculate the eigenvalues and eigenvectors *)
 {λ, ψ} = {λ[[#]], ψ[[#]]} &@Ordering[λ]; (* Sort them in ascending order *)
 ψ = MatrixNormalize[#, S] & /@ ψ;
 (* Normalize them w.r.t. the overlap matrix S *)
 (* Perform a numerical divergence check *)
 If[AllTrue[Thread[#.S.# ≠ 1] & /@ ψ, TrueQ], Abort[], Nothing];
 (* Calculate the density matrix from the
  eigenvector ψ_1 associated with the minimal eigenvalue *)
 P = 2 TensorProduct[ψ_1, ψ_1];
];

(* Return the first eigenvalue and the equilibrium bonding energy,
where 1/d is the E_nucl term *)
(* The `Total[#, 2]` function
 gives the sum of all elements of the matrix # *)
{λ_1, Total[P (H + 1/4 G), 2] + 1/d}
];
```

In[36]:=
```
δ = Range[0.3, 10, 0.01] // N;
ϵ = 10^-4 // N;(* Accuracy level *)

{τ, ρ} = ParallelMap[Quiet@SCFLoop[#, ϵ, α] &, δ] // Transpose // AbsoluteTiming;
```

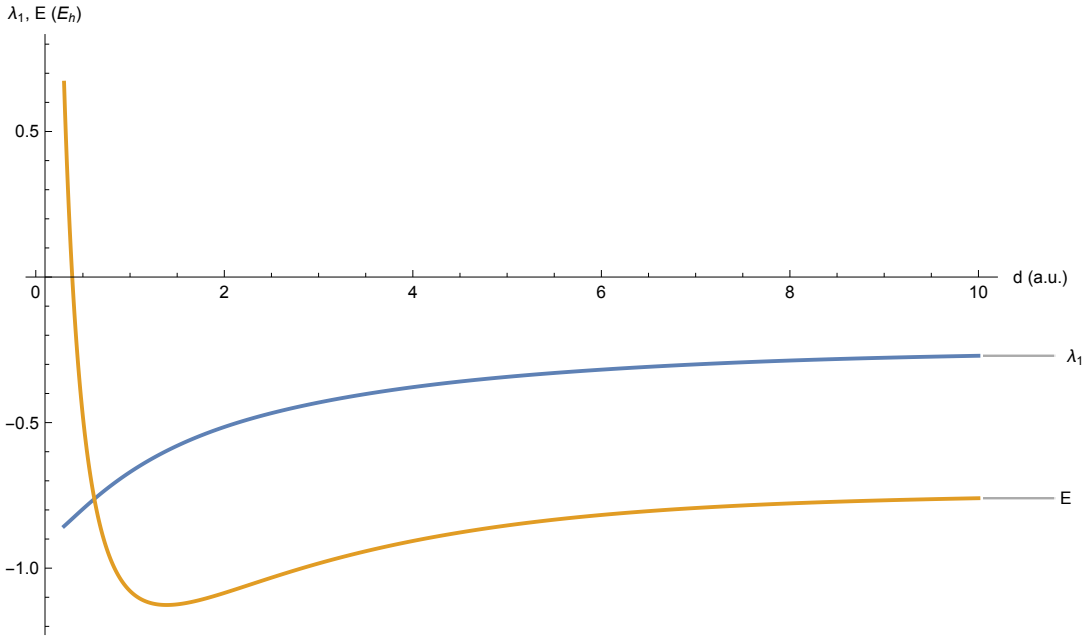
In[39]:=
```
Length@δ (* Number of distances used *)
Quantity[τ, "Seconds"]
(* Time it took the loop to finish, measured in seconds *)
```

Out[39]=
```
971
```

Out[40]=
```
8.97029 s
```

In[41]:=
```
(* Smallest eigenvalue λ1 and equilibrium bonding energy E *)
ListLinePlot[Thread[{δ, #}] & /@ ρ, AxesLabel → {"d (a.u.)", "λ1, E (Eh)"},
 PlotRange → Full, ImageSize → Large, PlotLabels → {"λ1", "E"}]
```

Out[41]=



In[42]:=
```
Quantity[Min[Last@ρ], "HartreeEnergy"] (* Smallest energy value*)
UnitConvert[%, "Electronvolts"](* ...converted in electronvolts *)
```

Out[42]=

$-1.12654\ E_h$

Out[43]=

-30.6548 eV