1 Introduction

We consider the one-electron Hydrogenic-atom Hamiltonian, which is of the form

$$\hat{H}\,|\psi\rangle = E\,|\psi\rangle$$

where $\hat{H} = \hat{K} + \hat{V}$, with

$$\hat{K}\,|\psi\rangle = \left[-\frac{1}{2r}\frac{\partial^2}{\partial r^2}(r\cdot) + \frac{1}{2r^2}\hat{L}^2\right]|\psi\rangle \quad \text{and} \quad \hat{V}\,|\psi\rangle = -\frac{Z}{r}\,|\psi\rangle$$

and where $\hat{L} = \hat{L}_x + \hat{L}_y + \hat{L}_z$ is the angular momentum operator, which has eigenstates $|Y_\ell^m\rangle$ which satisfy

$$\hat{L}^2\,|Y_\ell^m\rangle = \ell(\ell+1)\,|Y_\ell^m\rangle \quad \text{and} \quad \hat{L}_z\,|Y_\ell^m\rangle = m\,|Y_\ell^m\rangle\,.$$

We solve this system by the method of basis expansion, where we utilise a basis of the form, $\mathcal{B} = \{|\phi_i\rangle\}_{i=1}^N$ which we suppose to be complete in the limit as $N \to \infty$. We select the basis functions, represented in coordinate-space, to be of the form

$$\phi_i(r,\Omega) = \frac{1}{r}\varphi_{k_i,\ell_i}(r)Y_{\ell_i}^{m_i}(\Omega) \quad \text{for} \quad i = 1,\ldots,N$$

where the radial functions, $\mathcal{R} = \{|\varphi_{k_i,\ell_i}\rangle\}_{i=1}^N$ form a complete basis for the radial function space, in the limit as $N \to \infty$. For elements of this basis, the one-electron Hydrogenic-atom Hamiltonian assumes the form

$$\begin{aligned}
\hat{H}\,|\phi_i\rangle &= \left[-\frac{1}{2r}\frac{\partial^2}{\partial r^2}(r\cdot) + \frac{1}{2r^2}\hat{L}^2 - \frac{Z}{r}\right]|\phi_i\rangle \\
&= \left[-\frac{1}{2r}\frac{\partial^2}{\partial r^2}(r\cdot) + \frac{\ell_i(\ell_i+1)}{2r^2} - \frac{Z}{r}\right]|\phi_i\rangle \\
&= \left[-\frac{1}{2r}\frac{\partial^2}{\partial r^2}(r\cdot) + \frac{\ell_i(\ell_i+1)}{2r^2} - \frac{Z}{r}\right]\left|\tfrac{1}{r}\varphi_{k_i,\ell_i}, Y_{\ell_i}^{m_i}\right\rangle
\end{aligned}$$

thus reducing to operator which acts purely to radial terms, indexed by ℓ_i. Lastly, we note that the inner product is of the form

$$\langle\phi_i|\,\hat{A}\,|\phi_j\rangle = \int_0^\infty \mathrm{d}r\, r^2 \int_\Omega \mathrm{d}\Omega\, \overline{\phi_i(r,\Omega)}\hat{A}\big[\phi_j(r,\Omega)\big]$$

where $\hat{A}$ is an arbitrary linear operator, and whence, in the case where $\hat{A}$ can be reduced to an operator which acts only on radial terms, indexed by ℓ, we have that

$$\begin{aligned}
\langle\phi_i|\,\hat{A}\,|\phi_j\rangle &= \int_0^\infty \mathrm{d}r\, r^2 \overline{\tfrac{1}{r}\varphi_{k_i,\ell_i}(r)}\hat{A}_{\ell_j}\big[\tfrac{1}{r}\varphi_{k_j,\ell_j}(r)\big]\int_\Omega \mathrm{d}\Omega\, \overline{Y_{\ell_i}^{m_i}(\Omega)}Y_{\ell_j}^{m_j}(\Omega) \\
&= \int_0^\infty \mathrm{d}r\, r^2 \overline{\tfrac{1}{r}\varphi_{k_i,\ell_i}(r)}\hat{A}_{\ell_j}\big[\tfrac{1}{r}\varphi_{k_j,\ell_j}(r)\big]\delta_{\ell_i,\ell_j}\delta_{m_i,m_j} \\
&= \big\langle \tfrac{1}{r}\varphi_{k_i,\ell_i}\big|\,\hat{A}_{\ell_j}\,\big|\tfrac{1}{r}\varphi_{k_j,\ell_j}\big\rangle\,\delta_{\ell_i,\ell_j}\delta_{m_i,m_j}
\end{aligned}$$

where we have defined the radial inner product to be of the form

$$\big\langle \tfrac{1}{r}\varphi_{k_i,\ell_i}\big|\,\hat{A}_{\ell_j}\,\big|\tfrac{1}{r}\varphi_{k_j,\ell_j}\big\rangle = \int_0^\infty \mathrm{d}r\, r^2 \overline{\tfrac{1}{r}\varphi_{k_i,\ell_i}(r)}\hat{A}_{\ell_j}\big[\tfrac{1}{r}\varphi_{k_j,\ell_j}(r)\big].$$

2 Laguerre Basis

We utilise a Laguerre basis for the set of radial functions which, for $k = 1, 2, \ldots$ and where $\ell \in \{0, 1, \ldots\}$, are of the following form in coordinate-space

$$\varphi_{k,\ell}(r) = N_{k,\ell}(2\alpha_\ell r)^{\ell+1} \exp(-\alpha_\ell r) L_{k-1}^{2\ell+1}(2\alpha_\ell r)$$

where $\alpha_\ell \in (0, \infty)$ is an arbitrarily chosen constant, where $N_{k,\ell}$ are the normalisation constants, given by

$$N_{k,\ell} = \sqrt{\frac{\alpha_\ell (k-1)!}{(k+\ell)(k+2\ell)!}}$$

and where $L_{k-1}^{2\ell+1}$ are the generalised Laguerre polynomials.

2.1 Recurrence Relation

We construct the Laguerre basis by means of the following recurrence relation of the Laguerre polynomials

$$\begin{aligned} L_0^t(x) &= 1 \\ L_1^t(x) &= 1 + t - x \\ (n+1)L_{n+1}^t(x) &= (2n + 1 + t - x)L_n^t(x) - (n+t)L_{n-1}^t(x) \quad \text{for} \quad n = 1, 2, \ldots . \end{aligned}$$

Firstly, we write $\varphi_{k,\ell}(r) = N_{k,\ell}\widetilde{\varphi}_{k,\ell}(r)$, whence we note that

$$\begin{aligned} \widetilde{\varphi}_{1,\ell}(r) &= (2\alpha_\ell r)^{\ell+1} \exp(-\alpha_\ell r) \\ \widetilde{\varphi}_{2,\ell}(r) &= 2(\ell + 1 - \alpha_\ell r)(2\alpha_\ell r)^{\ell+1} \exp(-\alpha_\ell r) \\ (k-1)\widetilde{\varphi}_{k,\ell}(r) &= 2(k - 1 + \ell - \alpha_\ell r)\widetilde{\varphi}_{k-1,\ell}(r) - (k + 2\ell - 1)\widetilde{\varphi}_{k-2,\ell}(r) \quad \text{for} \quad k = 3, 4, \ldots , \end{aligned}$$

from which can trivially recover the functions $\varphi_{k,\ell}(r)$.

2.2 Normalisation Constant Recurrence Relation

To circumvent overflow errors when calculating the normalisation constant, $N_{k,\ell}$, we construct these constants using a recurrence relations. We note that

$$\begin{aligned} N_{k,\ell} &= \sqrt{\frac{\alpha_\ell (k-1)!}{(k+\ell)(k+2\ell)!}} \\ &= \sqrt{\frac{(k-1)(k-1+\ell)}{(k+\ell)(k+2\ell)} \frac{\alpha_\ell (k-2)!}{(k-1+\ell)(k+2\ell-1)!}} \\ &= \sqrt{\frac{(k-1)(k-1+\ell)}{(k+\ell)(k+2\ell)}} N_{k-1,\ell} \end{aligned}$$

for $k = 2, 3, \ldots$ and where $\ell \in \{0, 1, \ldots\}$, and that

$$N_{1,\ell} = \sqrt{\frac{\alpha_\ell}{(\ell+1)(2\ell+1)!}}$$

yielding a numerically-stable recurrence relation for the normalisation constants as required.

2.3 Laguerre Radial Basis Code

FORTRAN code for calculating the Laguerre basis functions for a given radial grid can be found in src/laguerre.f90: `subroutine radial_basis()`, and is shown in Listing 1.

```
7   ! radial_basis
8   !
9   ! phi_{k, l, m}(r, theta, phi) = (varphi_{k, l}(r) / r) * Y_{l, m}(theta, phi)
10  ! where
11  ! varphi_{k, l}(r) = sqrt(alpha * (k - 1)! / (k + l) * (k + 2*l)!)
12  !                     * (2*alpha*r)^{l+1}
13  !                     * exp(-alpha*r)
14  !                     * L_{k - 1}^{2*l + 1}(2*alpha*r)
15  ! where L_{i}^{j} are the generalised Laguerre polynomials.
16  !
17  ! For given l, alpha, and r_grid, yields the functions varphi_{k, l}(r) for
18  ! k = 1, ..., n_basis, on the radial values specified in the grid.
19  !
20  ! Also returns an error code <ierr> where:
21  ! - 0 indicates successful execution,
22  ! - 1 indicates invalid arguments.
23  pure subroutine radial_basis (l, alpha, n_r, r_grid, n_basis, basis, ierr)
24    integer , intent(in) :: l, n_r, n_basis
25    double precision , intent(in) :: alpha
26    double precision , intent(in) :: r_grid(n_r)
27    double precision , intent(out) :: basis(n_r, n_basis)
28    integer , intent(out) :: ierr
29    double precision :: norm(n_basis)
30    double precision :: alpha_grid(n_r)
31    integer :: kk
32
33    ! check if arguments are valid
34    ierr = 0
35
36    if ((l < 0) .or. (n_basis < 1) .or. (n_r < 1)) then
37      ierr = 1
38      return
39    end if
40
41    ! recurrence relation for basis normalisation constants
42    norm(1) = sqrt(alpha / dble((l + 1) * gamma(dble((2 * l) + 2))))
43
44    if (n_basis >= 2) then
45      do kk = 2, n_basis
46        norm(kk) = norm(kk-1) * sqrt(dble((kk - 1) * (kk - 1 + l)) / &
47            dble((kk + l) * (kk + (2 * l))))
48      end do
49    end if
50
51    ! in-lined array since r_grid(:) on its own is never used
52    alpha_grid(:) = alpha * r_grid(:)
53
54    ! recurrence relation for basis functions
55    basis(:, 1) = ((2.0d0 * alpha_grid(:)) ** (l + 1)) * &
56        exp(-alpha_grid(:))
57
58    if (n_basis >= 2) then
59      basis(:, 2) = 2.0d0 * (dble(l + 1) - alpha_grid(:)) * basis(:, 1)
```

```
60      end if
61
62      if (n_basis >= 3) then
63        do kk = 3, n_basis
64          basis(:, kk) = &
65              ((2.0d0 * (dble(kk - 1 + l) - alpha_grid(:)) * basis(:, kk-1)) &
66              - dble(kk + (2 * l) - 1) * basis(:, kk-2)) / dble(kk - 1)
67        end do
68      end if
69
70      ! scaling basis functions by normalisation constants
71      do kk = 1, n_basis
72        basis(:, kk) = basis(:, kk) * norm(kk)
73      end do
74
75    end subroutine radial_basis
```

Listing 1: Calculation of Laguerre radial basis functions for a given radial grid.

2.4 Laguerre Radial Basis Figures

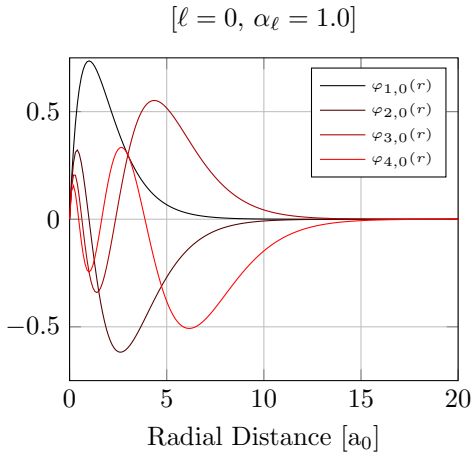
A radial grid has been constructed, for given $\mathrm{d_r}$ and $\mathrm{r_{max}}$, of the form

$$\{r_i = \mathrm{d_r} \cdot (i-1)\}_{i=1}^{\mathrm{n_r}}$$

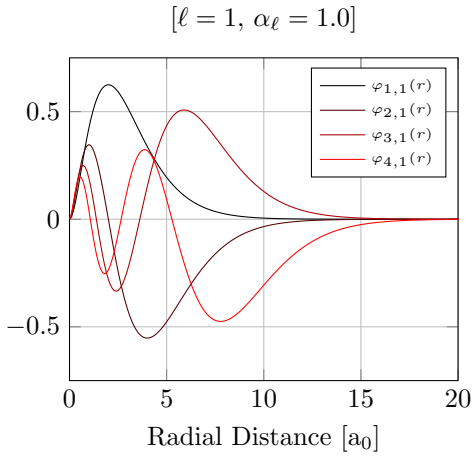
where $\mathrm{n_r}$ is the smallest integer such that

$$\mathrm{d_r} \cdot (\mathrm{n_r} - 1) \geq \mathrm{r_{max}}.$$

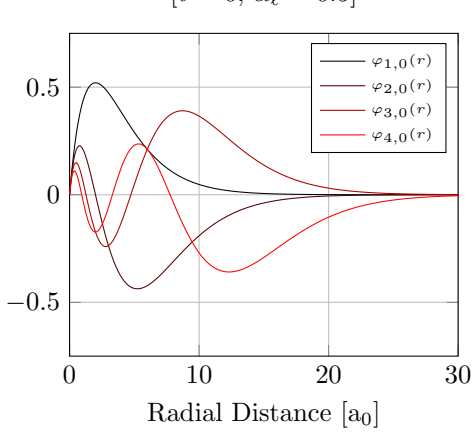
The Laguerre basis functions have been calculated on this radial grid, for various values of ℓ and α_ℓ. The plots of the first 4 basis functions for these values of ℓ and α_ℓ are shown in Figure 1.
It can be seen from Figure 1 that as ℓ increases, the Laguerre radial functions are somewhat shifted further from the origin; that is, they are suppressed at the origin, peak at a further distance away from the origin, and extend further away from the origin before exponentially decaying to 0. They also have wider and less pronounced peaks.
It can also be seen from Figure 1 that as α_ℓ decreases, the Laguerre radial basis functions extend much further away from the origin before exponentially decaying to 0, and have wider and less pronounced peaks.



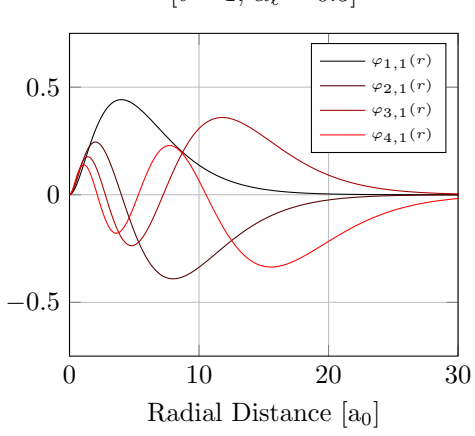
(a) $\varphi_{k,0}(r)$ *for* $k = 1, 2, 3, 4$, *with* $\alpha_\ell = 1.0$.



(b) $\varphi_{k,1}(r)$ *for* $k = 1, 2, 3, 4$, *with* $\alpha_\ell = 1.0$.



(c) $\varphi_{k,0}(r)$ *for* $k = 1, 2, 3, 4$, *with* $\alpha_\ell = 0.5$.



(d) $\varphi_{k,1}(r)$ *for* $k = 1, 2, 3, 4$, *with* $\alpha_\ell = 0.5$.

Figure 1: The first four Laguerre radial basis functions are plotted for various cases of ℓ *and* α_ℓ. *Note that every figure has the same y-axes bounds* $[-0.75, 0.75]$, *whereas the x-axes bounds are* $[0, 20]$ *for the* $\alpha_\ell = 1.0$ *cases, and* $\{0, 30\}$ *for the* $\alpha_\ell = 0.5$ *cases. Observe that* $\varphi_{k,\ell}(r)$ *has* k *extremal points, with each extrema being larger (in magnitude) than the preceeding extrema, before eventually exhibiting exponential decay to 0, after the last extremal point.*

3 Kinetic Energy Matrix Elements

Here, we shall derive an analytic expression for the kinetic energy matrix elements, with regard to the Laguerre radial basis. Firstly, we note that the kinetic energy matrix elements are defined by

$$K_{i,j} = \langle\phi_i| \hat{K} |\phi_j\rangle = \langle\phi_i| \left[-\frac{1}{2r}\frac{\partial^2}{\partial r^2}(r\cdot) + \frac{1}{2r^2}\hat{L}^2\right] |\phi_j\rangle$$

whence we note that

$$\begin{aligned} K_{i,j} &= \left\langle \tfrac{1}{r}\varphi_{k_i,\ell_i}, Y_{\ell_i}^{m_i}\right| \left[-\frac{1}{2r}\frac{\partial^2}{\partial r^2}(r\cdot) + \frac{1}{2r^2}\hat{L}^2\right] \left|\tfrac{1}{r}\varphi_{k_j,\ell_j}, Y_{\ell_j}^{m_j}\right\rangle \\ &= \left\langle \tfrac{1}{r}\varphi_{k_i,\ell_i}\right| \left[-\frac{1}{2r}\frac{\mathrm{d}^2}{\mathrm{d}r^2}(r\cdot) + \frac{\ell_j(\ell_j+1)}{2r^2}\right] \left|\tfrac{1}{r}\varphi_{k_j,\ell_j}\right\rangle \left\langle Y_{\ell_i}^{m_i}\middle| Y_{\ell_j}^{m_j}\right\rangle \\ &= \left\langle \tfrac{1}{r}\varphi_{k_i,\ell_i}\right| \hat{K}_{\ell_j} \left|\tfrac{1}{r}\varphi_{k_j,\ell_j}\right\rangle \delta_{\ell_i,\ell_j}\delta_{m_i,m_j}. \end{aligned}$$

We note that since the matrix element is necessarily zero, where $\ell_i \neq \ell_j$, we restrict our attention to the case where $\ell_i = \ell_j = \ell$. It follows that the radial terms can be written in the form

$$\begin{aligned} \left\langle \tfrac{1}{r}\varphi_{k_i,\ell}\right| \hat{K}_\ell \left|\tfrac{1}{r}\varphi_{k_j,\ell}\right\rangle &= \left\langle \tfrac{1}{r}\varphi_{k_i,\ell}\right| \left[-\frac{1}{2r}\frac{\mathrm{d}^2}{\mathrm{d}r^2}(r\cdot) + \frac{\ell(\ell+1)}{2r^2}\right] \left|\tfrac{1}{r}\varphi_{k_j,\ell}\right\rangle \\ &= \int_0^\infty \mathrm{d}r\, r^2 \overline{\tfrac{1}{r}\varphi_{k_i,\ell}(r)} \left[-\frac{1}{2r}\frac{\mathrm{d}^2}{\mathrm{d}r^2}(r\cdot) + \frac{\ell(\ell+1)}{2r^2}\right] \left(\tfrac{1}{r}\varphi_{k_j,\ell}(r)\right) \\ &= \int_0^\infty \mathrm{d}r\, \varphi_{k_i,\ell}(r) \left[-\frac{1}{2}\frac{\mathrm{d}^2}{\mathrm{d}r^2} + \frac{\ell(\ell+1)}{2r^2}\right] \varphi_{k_j,\ell}(r) \end{aligned}$$

where we have dropped the conjugacy due to the Laguerre radial basis functions being entirely real-valued. Expanding this fully, we have that

$$\begin{aligned} \left\langle \tfrac{1}{r}\varphi_{k_i,\ell}\right| \hat{K}_\ell \left|\tfrac{1}{r}\varphi_{k_j,\ell}\right\rangle &= \int_0^\infty \mathrm{d}r \left(N_{k_i,\ell}(2\alpha r)^{\ell+1}\exp(-\alpha r) L_{k_i-1}^{2\ell+1}(2\alpha r)\right) \\ &\quad \times \left[-\frac{1}{2}\frac{\mathrm{d}^2}{\mathrm{d}r^2} + \frac{\ell(\ell+1)}{2r^2}\right] \left(N_{k_j,\ell}(2\alpha r)^{\ell+1}\exp(-\alpha r) L_{k_j-1}^{2\ell+1}(2\alpha r)\right) \end{aligned}$$

whence we introduce the variable transformation $x = 2\alpha r$, to yield an equivalent integral of the form

$$\begin{aligned} \left\langle \tfrac{1}{r}\varphi_{k_i,\ell}\right| \hat{K}_\ell \left|\tfrac{1}{r}\varphi_{k_j,\ell}\right\rangle &= (2\alpha) N_{k_i,\ell} N_{k_j,\ell} \int_0^\infty \mathrm{d}x\, x^{\ell+1}\exp(-\tfrac{x}{2}) L_{k_i-1}^{2\ell+1}(x) \\ &\quad \times \left[-\frac{1}{2}\frac{\mathrm{d}^2}{\mathrm{d}x^2} + \frac{\ell(\ell+1)}{2x^2}\right] \left(x^{\ell+1}\exp(-\tfrac{x}{2}) L_{k_j-1}^{2\ell+1}(x)\right). \end{aligned}$$

At this point, we note that

$$\begin{aligned} &\frac{\mathrm{d}^2}{\mathrm{d}x^2}\left(x^{\ell+1}\exp(-\tfrac{x}{2}) L_{k_j-1}^{2\ell+1}(x)\right) = x^{\ell+1}\exp(-\tfrac{x}{2}) \\ &\quad \times \left(\left(\frac{\ell(\ell+1)}{x^2} - \frac{\ell+1}{x} + \frac{1}{4}\right) L_{k_j-1}^{2\ell+1}(x) + \left(\frac{2(\ell+1)}{x} - 1\right)\frac{\mathrm{d}}{\mathrm{d}x}\left(L_{k_j-1}^{2\ell+1}(x)\right) + \frac{\mathrm{d}^2}{\mathrm{d}x^2}\left(L_{k_j-1}^{2\ell+1}(x)\right)\right) \end{aligned}$$

whence

$$\left[-\frac{1}{2}\frac{\mathrm{d}^2}{\mathrm{d}x^2}+\frac{\ell(\ell+1)}{2x^2}\right]\left(x^{\ell+1}\exp(-\tfrac{x}{2})L_{k_j-1}^{2\ell+1}(x)\right) = -\frac{1}{2}x^{\ell+1}\exp(-\tfrac{x}{2}) \times\left(\left(-\frac{\ell+1}{x}+\frac{1}{4}\right)L_{k_j-1}^{2\ell+1}(x)+\left(\frac{2(\ell+1)}{x}-1\right)\frac{\mathrm{d}}{\mathrm{d}x}\left(L_{k_j-1}^{2\ell+1}(x)\right)+\frac{\mathrm{d}^2}{\mathrm{d}x^2}\left(L_{k_j-1}^{2\ell+1}(x)\right)\right).$$

We utilise the following recurrence relation of the generalised Laguerre polynomials,

$$\frac{t+1-x}{x}\frac{\mathrm{d}}{\mathrm{d}x}\left(L_n^t(x)\right)+\frac{\mathrm{d}^2}{\mathrm{d}x^2}\left(L_n^t(x)\right)=-\frac{n}{x}L_n^t(x)$$

to further simplify the above term to the form

$$\left[-\frac{1}{2}\frac{\mathrm{d}^2}{\mathrm{d}x^2}+\frac{\ell(\ell+1)}{2x^2}\right]\left(x^{\ell+1}\exp(-\tfrac{x}{2})L_{k_j-1}^{2\ell+1}(x)\right)=\frac{1}{2}\left(\frac{k_j+\ell}{x}-\frac{1}{4}\right)x^{\ell+1}\exp(-\tfrac{x}{2})L_{k_j-1}^{2\ell+1}(x)$$

whence the integral becomes

$$\left\langle\tfrac{1}{r}\varphi_{k_i,\ell}\right|\hat{K}_\ell\left|\tfrac{1}{r}\varphi_{k_j,\ell}\right\rangle=\alpha N_{k_i,\ell}N_{k_j,\ell}\int_0^\infty\mathrm{d}x\left(k_j+\ell-\frac{x}{4}\right)x^{2\ell+1}\exp(-x)L_{k_i-1}^{2\ell+1}(x)L_{k_j-1}^{2\ell+1}(x)$$

We note that

$$\left\langle\tfrac{1}{r}\varphi_{k_i,\ell}\right|\left.\tfrac{1}{r}\varphi_{k_j,\ell}\right\rangle=\frac{N_{k_i,\ell}N_{k_j,\ell}}{2\alpha}\int_0^\infty\mathrm{d}x\,x^{2\ell+2}\exp(-x)L_{k_i-1}^{2\ell+1}(x)L_{k_j-1}^{2\ell+1}(x)$$

whence the previous integral can be separated as

$$\begin{aligned}\left\langle\tfrac{1}{r}\varphi_{k_i,\ell}\right|\hat{K}_\ell\left|\tfrac{1}{r}\varphi_{k_j,\ell}\right\rangle &= \alpha N_{k_i,\ell}N_{k_j,\ell}(k_j+\ell)\int_0^\infty\mathrm{d}x\,x^{2\ell+1}\exp(-x)L_{k_i-1}^{2\ell+1}(x)L_{k_j-1}^{2\ell+1}(x)\\ &\quad-\frac{\alpha}{4}N_{k_i,\ell}N_{k_j,\ell}\int_0^\infty x^{2\ell+2}\exp(-x)L_{k_i-1}^{2\ell+1}(x)L_{k_j-1}^{2\ell+1}(x)\\ &= \alpha N_{k_i,\ell}N_{k_j,\ell}(k_j+\ell)\int_0^\infty\mathrm{d}x\,x^{2\ell+1}\exp(-x)L_{k_i-1}^{2\ell+1}(x)L_{k_j-1}^{2\ell+1}(x)\\ &\quad-\frac{\alpha^2}{2}\left\langle\tfrac{1}{r}\varphi_{k_i,\ell}\right|\left.\tfrac{1}{r}\varphi_{k_j,\ell}\right\rangle.\end{aligned}$$

At this point we note the following property of the generalised Laguerre polynomials,

$$\int_0^\infty\mathrm{d}x\,x^t\exp(-x)L_n^t(x)L_m^t(x)=\frac{(n+t)!}{n!}\delta_{m,n}$$

whence the radial term of the kinetic energy matrix elements is shown to be given analytically by the expression

$$\begin{aligned}\left\langle\tfrac{1}{r}\varphi_{k_i,\ell}\right|\hat{K}_\ell\left|\tfrac{1}{r}\varphi_{k_j,\ell}\right\rangle &= \alpha N_{k_i,\ell}^2(k_j+\ell)\frac{(k_j+2\ell)!}{(k_j-1)!}\delta_{k_i,k_j}-\frac{\alpha^2}{2}\left\langle\tfrac{1}{r}\varphi_{k_i,\ell}\right|\left.\tfrac{1}{r}\varphi_{k_j,\ell}\right\rangle\\ &= \alpha^2\delta_{k_i,k_j}-\frac{\alpha^2}{2}\left\langle\tfrac{1}{r}\varphi_{k_i,\ell}\right|\left.\tfrac{1}{r}\varphi_{k_j,\ell}\right\rangle.\end{aligned}$$

It follows that the kinetic energy matrix elements are thus of the form

$$K_{i,j}=\alpha^2\left(\delta_{k_i,k_j}-\frac{1}{2}\left\langle\tfrac{1}{r}\varphi_{k_i,\ell}\right|\left.\tfrac{1}{r}\varphi_{k_j,\ell}\right\rangle\right)\delta_{\ell_i,\ell_j}\delta_{m_i,m_j}.$$

3.1 Extension: Overlap Matrix Elements

4 Atomic Hydrogen States

4.1 Hydrogenic Atom Code

4.1.1 Overlap Matrix Elements

FORTRAN code for calculating the overlap matrix elements for a Laguerre radial basis of a given dimension can be found in src/laguerre.f90: `subroutine overlap_matrix()`, and is shown in Listing 2.

```
77   ! overlap_matrix
78   !
79   ! < phi_{k', l, m} | phi_{k, l, m} >
80   !
81   ! Overlap matrix elements for given l, m.
82   ! We can restrict our attention to considering fixed l and m, since the matrix
83   ! elements are zero when l' /= l or where m' /= m.
84   ! Furthermore, the exponential decay variable, alpha, has no influence on
85   ! these matrix elements, nor does the magnetic quantum number, m.
86   !
87   ! Also returns an error code <ierr> where:
88   ! - 0 indicates successful execution,
89   ! - 1 indicates invalid arguments.
90   pure subroutine overlap_matrix(l, n_basis, B, ierr)
91     integer , intent(in) :: l, n_basis
92     double precision , intent(out) :: B(n_basis, n_basis)
93     integer , intent(out) :: ierr
94     integer :: kk
95
96     ! check if arguments are valid
97     ierr = 0
98
99     if ((l < 0) .or. (n_basis < 1)) then
100      ierr = 1
101      return
102    end if
103
104    ! initialise overlap matrix to zero
105    B(:, :) = 0.0d0
106
107    ! determine tri-diagonal overlap matrix elements
108    do kk = 1, n_basis-1
109      B(kk, kk) = 1.0d0
110
111      B(kk, kk+1) = - 0.5d0 * sqrt(1 - &
112          (dble(l * (l + 1)) / dble((kk + l) * (kk + l + 1))))
113
114      B(kk+1, kk) = B(kk, kk+1)
115    end do
116
117    ! last term (not covered by loop)
118    B(n_basis, n_basis) = 1.0d0
119
120  end subroutine overlap_matrix
```

Listing 2: Calculation of overlap matrix elements for a Laguerre radial basis of a given dimension.

4.1.2 Kinetic Energy Matrix Elements

FORTRAN code for calculating the kinetic energy matrix elements for a Laguerre radial basis of a given dimension can be found in src/laguerre.f90: `subroutine kinetic_matrix()`, and is shown in Listing 3.

```
122   ! kinetic_matrix
123   !
124   ! < phi_{k', l, m} | K | phi_{k, l, m} >
125   !
126   ! Kinetic matrix elements for given l, m, alpha.
127   ! We can restrict our attention to considering fixed l and m, since the matrix
128   ! elements are zero when l' /= l or where m' /= m.
129   ! Furthermore, the magnetic quantum number, m, has no influence on these
130   ! matrix elements.
131   !
132   ! Also returns an error code <ierr> where:
133   ! - 0 indicates successful execution,
134   ! - 1 indicates invalid arguments.
135   pure subroutine kinetic_matrix(l, alpha, n_basis, K, ierr)
136     integer , intent(in) :: l, n_basis
137     double precision , intent(in) :: alpha
138     double precision , intent(out) :: K(n_basis, n_basis)
139     integer , intent(out) :: ierr
140     integer :: kk
141
142     ! check if arguments are valid
143     ierr = 0
144
145     if ((l < 0) .or. (n_basis < 1)) then
146       ierr = 1
147       return
148     end if
149
150     ! initialise kinetic matrix to zero
151     K(:, :) = 0.0d0
152
153     ! determine tri-diagonal kinetic matrix elements
154     do kk = 1, n_basis-1
155       K(kk, kk) = 0.5d0 * (alpha ** 2)
156
157       K(kk, kk+1) = (alpha ** 2) * 0.25d0 * sqrt(1 - &
158           (dble(l * (l + 1)) / dble((kk + l) * (kk + l + 1))))
159
160       K(kk+1, kk) = K(kk, kk+1)
161     end do
162
163     ! last term (not covered by loop)
164     K(n_basis, n_basis) = 0.5d0 * (alpha ** 2)
165
166   end subroutine kinetic_matrix
```

Listing 3: Calculation of kinetic energy matrix elements for a Laguerre radial basis of a given dimension.

4.1.3 Coulomb Potential Matrix Elements

FORTRAN code for calculating the Coulomb potential matrix elements for a Laguerre radial basis of a given dimension can be found in src/laguerre.f90: subroutine coulomb_matrix(), and is shown in Listing 4.

```
168   ! coulomb_matrix
169   !
170   ! < phi_{k', l, m} | 1/r | phi_{k, l, m} >
171   !
172   ! Coulomb matrix elements for given l, m, alpha.
173   ! We can restrict our attention to considering fixed l and m, since the matrix
174   ! elements are zero when l' /= l or where m' /= m.
175   ! Furthermore, the magnetic quantum number, m, has no influence on these
176   ! matrix elements.
177   !
178   ! Also returns an error code <ierr> where:
179   ! - 0 indicates successful execution,
180   ! - 1 indicates invalid arguments.
181   pure subroutine coulomb_matrix(l, alpha, n_basis, V, ierr)
182     integer , intent(in) :: l, n_basis
183     double precision , intent(in) :: alpha
184     double precision , intent(out) :: V(n_basis, n_basis)
185     integer , intent(out) :: ierr
186     integer :: kk
187
188     ! check if arguments are valid
189     ierr = 0
190
191     if ((l < 0) .or. (n_basis < 1)) then
192       ierr = 1
193       return
194     end if
195
196     ! initialise coulomb matrix to zero
197     V(:, :) = 0.0d0
198
199     ! determine diagonal coulomb matrix elements
200     do kk = 1, n_basis
201       V(kk, kk) = alpha / dble(kk + l)
202     end do
203
204   end subroutine coulomb_matrix
```

Listing 4: Calculation of Coulomb potential matrix elements for a Laguerre radial basis of a given dimension.

4.1.4 Hamiltonian Matrix Elements

FORTRAN code for calculating the overlap, kinetic energy, potential energy, and Hamiltonian matrix elements for a Laguerre radial basis of a given dimension can be found in src/laguerre.f90: subroutine hydrogenic_matrices(), and is shown in Listing 5.

```
206   ! hydrogenic_matrices
207   !
208   ! Yields overlap, kinetic, potential and Hamiltonian matrices for given l, m,
209   ! alpha, atomic_charge; that is: B, K, V, H.
```

```
210   ! We can restrict our attention to considering fixed l and m, since the matrix
211   ! elements are zero when l' /= l or where m' /= m.
212   ! Furthermore, the magnetic quantum number, m, has no influence on these
213   ! matrix elements.
214   !
215   ! Also returns an error code <ierr> where:
216   ! - 0 indicates successful execution,
217   ! - 1 indicates invalid arguments.
218   pure subroutine hydrogenic_matrices(l, alpha, atomic_charge, n_basis, B, &
219       K, V, H, ierr)
220     integer , intent(in) :: l, atomic_charge, n_basis
221     double precision , intent(in) :: alpha
222     double precision , intent(out) :: B(n_basis, n_basis), K(n_basis, n_basis), &
223         V(n_basis, n_basis), H(n_basis, n_basis)
224     integer , intent(out) :: ierr
225
226     ! check if arguments are valid
227     ierr = 0
228
229     if ((l < 0) .or. (n_basis < 1)) then
230       ierr = 1
231       return
232     end if
233
234     ! calculate matrices
235     call overlap_matrix(l, n_basis, B, ierr)
236
237     call kinetic_matrix(l, alpha, n_basis, K, ierr)
238
239     call coulomb_matrix(l, alpha, n_basis, V, ierr)
240     V(:, :) = - dble(atomic_charge) * V(:, :)
241
242     H(:, :) = K(:, :) + V(:, :)
243
244   end subroutine hydrogenic_matrices
```

Listing 5: Calculation of overlap, kinetic energy, potential energy, and Hamiltonian matrix elements for a Laguerre radial basis of a given dimension.

4.1.5 Hydrogenic Atom Program

A FORTRAN program which calculates the pseudo-energies and pseudo-states of a one-electron Hydrogenic-atom Hamiltonian by the method of basis expansion, using the Laguerre radial basis, for given $(\ell, m, \alpha_\ell, Z, \mathrm{n_{basis}}, \mathrm{d_r}, \mathrm{r_{max}})$, can be found in `src/hydrogenic_atom.f90` and is shown in Listing 6.

```
1  !>
2  program hydrogenic_atom
3
4    use laguerre
5    use io
6
7    implicit none
8
9    ! global flags
10   logical , parameter :: debugging = .true.
```

```
11    logical , parameter :: display_bases = .false.
12    logical , parameter :: display_matrices = .true.
13
14    ! angular quantum number variables
15    integer :: l, m
16    double precision :: alpha
17
18    ! atomic variables
19    integer :: atomic_charge
20
21    ! radial grid variables
22    integer :: n_r
23    double precision :: d_r, r_max
24    double precision , allocatable :: r_grid(:)
25
26    ! basis variables
27    integer :: n_basis
28    double precision , allocatable :: basis(:, :)
29    double precision , allocatable :: eigen_basis(:, :)
30
31    ! matrices
32    double precision , allocatable :: B(:, :), K(:, :), V(:, :), H(:, :)
33    double precision , allocatable :: eigen_values(:), eigen_vectors(:, :)
34
35    ! local variables
36    integer :: ierr
37    integer :: ii
38
39    ! read parameters from command line arguments
40    call read_input(l, m, alpha, atomic_charge, n_basis, d_r, r_max)
41
42    ! check if parameters are valid
43    if ((l < 0) .or. (abs(m) > l) &
44        .or. (alpha < 0.0d0) &
45        .or. (n_basis < 1) &
46        .or. (d_r < 0.0d0) .or. (r_max < 0.0d0)) then
47      ierr = 1
48      write (*, *) "parameters are invalid, failing with <ierr>: ", ierr
49      call exit(ierr)
50    end if
51
52    ! set n_r
53    n_r = ceiling(r_max / d_r) + 1
54
55    ! allocate arrays
56    allocate(r_grid(n_r))
57
58    allocate(basis(n_r, n_basis))
59    allocate(eigen_basis(n_r, n_basis))
60
61    allocate(B(n_basis, n_basis))
62    allocate(K(n_basis, n_basis))
63    allocate(V(n_basis, n_basis))
64    allocate(H(n_basis, n_basis))
65
66    allocate(eigen_values(n_basis))
67    allocate(eigen_vectors(n_basis, n_basis))
68
```

```
69    ! initialise radial grid
70    do ii = 1, n_r
71      r_grid(ii) = d_r * (ii - 1)
72    end do
73
74    ! calculate radial basis functions
75    call radial_basis(l, alpha, n_r, r_grid, n_basis, basis, ierr)
76
77    if (ierr /= 0) then
78      write (*, *) "radial_basis() failed with <ierr>: ", ierr
79      call exit(ierr)
80    end if
81
82    if (display_bases) then
83      write (*, *) "basis(n_r, n_basis)"
84      call display_basis(n_r, r_grid, n_basis, basis)
85    end if
86
87    ! calculate matrix elements
88    call hydrogenic_matrices(l, alpha, atomic_charge, n_basis, B, K, V, H, ierr)
89
90    if (ierr /= 0) then
91      write (*, *) "hydrogenic_matrices() failed with <ierr>: ", ierr
92      call exit(ierr)
93    end if
94
95    if (display_matrices) then
96      write (*, *) "B(n_basis, n_basis)"
97      call display_matrix(n_basis, n_basis, B)
98
99      write (*, *) "K(n_basis, n_basis)"
100     call display_matrix(n_basis, n_basis, K)
101
102     write (*, *) "V(n_basis, n_basis)"
103     call display_matrix(n_basis, n_basis, V)
104
105     write (*, *) "H(n_basis, n_basis)"
106     call display_matrix(n_basis, n_basis, H)
107   end if
108
109   ! diagonalise
110   call diagonalise(n_r, n_basis, basis, B, H, eigen_values, eigen_vectors, &
111       eigen_basis)
112
113   if (display_matrices) then
114     write (*, *) "eigen_values(n_basis)"
115     call display_vector(n_basis, eigen_values)
116
117     write (*, *) "eigen_vectors(n_basis, n_basis)"
118     call display_matrix(n_basis, n_basis, eigen_vectors)
119   end if
120
121   if (display_bases) then
122     write (*, *) "eigen_basis(n_r, n_basis)"
123     call display_basis(n_r, r_grid, n_basis, eigen_basis)
124   end if
125
126   ! write output to file
```

```
127   call write_output(l, m, alpha, atomic_charge, n_r, r_grid, n_basis, basis, &
128       B, K, V, H, eigen_values, eigen_vectors, eigen_basis)
129
130   ! deallocate arrays
131   deallocate(r_grid)
132
133   deallocate(basis)
134   deallocate(eigen_basis)
135
136   deallocate(B)
137   deallocate(K)
138   deallocate(V)
139   deallocate(H)
140
141   deallocate(eigen_values)
142   deallocate(eigen_vectors)
143
144 contains
145
146   ! diagonalise
147   ! Note that since the call to rsg modifies the matrices it is given, we send
148   ! it copies of B, H.
149   subroutine diagonalise (n_r, n_basis, basis, B, H, eigen_values, &
150       eigen_vectors, eigen_basis)
151     integer , intent(in) :: n_r, n_basis
152     double precision , intent(in) :: basis(n_r, n_basis)
153     double precision , intent(in) :: B(n_basis, n_basis), H(n_basis, n_basis)
154     double precision , intent(out) :: eigen_values(n_basis)
155     double precision , intent(out) :: eigen_vectors(n_basis, n_basis)
156     double precision , intent(out) :: eigen_basis(n_r, n_basis)
157     double precision :: B_copy(n_basis, n_basis), H_copy(n_basis, n_basis)
158     integer :: ii, jj, kk
159     integer :: ierr
160     double precision :: temp_sum
161
162     ! create copies of B, H matrices to use in call to rsg subroutine
163     B_copy(:, :) = B(:, :)
164     H_copy(:, :) = H(:, :)
165
166     ! solve eigenvalue matrix equation
167     eigen_values(:) = 0.0d0
168     eigen_vectors(:, :) = 0.0d0
169
170     call rsg(n_basis, n_basis, H_copy, B_copy, eigen_values, 1, eigen_vectors, &
171         ierr)
172
173     if (ierr /= 0) then
174       write (*, *) "rsg() failed with <ierr>: ", ierr
175       call exit(ierr)
176     end if
177
178     ! calculate eigen-basis
179     eigen_basis(:, :) = 0.0d0
180
181     do jj = 1, n_basis
182       do ii = 1, n_r
183         temp_sum = 0.0d0
184
```

```
185         do kk = 1, n_basis
186           temp_sum = temp_sum + (eigen_vectors(kk, jj) * basis(ii, kk))
187         end do
188
189         eigen_basis(ii, jj) = temp_sum
190       end do
191     end do
192
193   end subroutine diagonalise
194
195 end program hydrogenic_atom
```

Listing 6: Calculation of pseudo-energies and pseudo-states of a one-electron Hydrogenic-atom Hamiltonian by the method of basis expansion, using the Laguerre radial basis.

4.2 Extension: He^+ Ion

4.3 Extension: Surface Plot in xz Plane

4.4 Extension: Numerically Calculating Potential Matrix Elements