Computer-based Exercises in Physical Chemistry

Raghunathan Ramakrishnan
ramakrishnan@tifrh.res.in

Tata Institute of Fundamental Research
TIFR Center for Interdisciplinary Sciences
Hyderabad, INDIA

26 December 2021

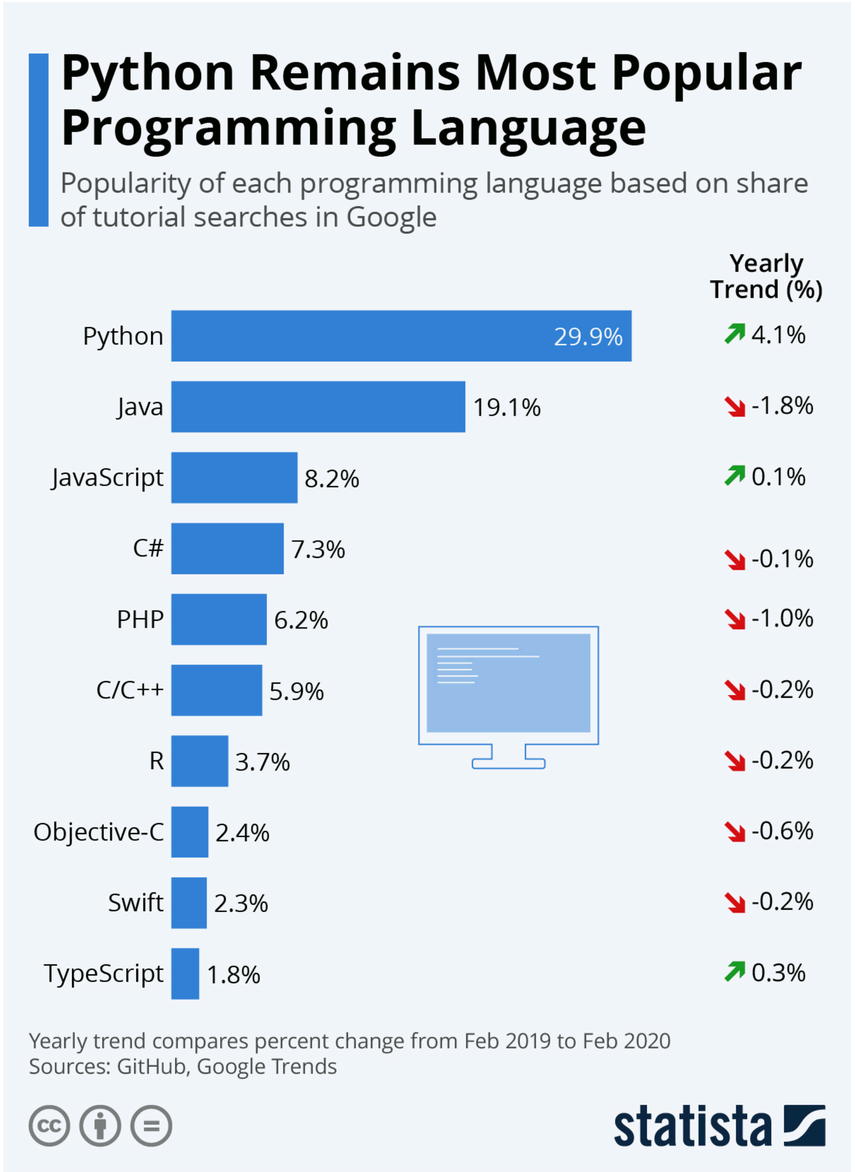
TIFR Centre for
Interdisciplinary
Studies

The presentation and the codes are available for download at https://github.com/raghurama123/Comp_PhysChem_Basic

Why should I learn about computers?

- Computers can be used as calculators, plotters, and notebooks.
- Parts of a program written for one specific task can be used for another task.
- Internet contains a lot of programs already written for chemistry tasks—from text book problems to research problems
- One can perform virtual experiments (simulations, modelling, etc.) that may not even be feasible to perform in a laboratory
- Visualization of concepts is possible using plotting libraries
- Extremely complicated mathematical equations can be solved very quickly using numerical methods

Choosing a programming language



Q: What's the best programming language to learn for science student with no previous programming experience

A: Python

Python is

- free
- easy to reference in the internet
- has a lot of libraries for visualisation, numerical methods, and data-analysis
- more libraries means less coding effort *so that one can focus on the research problem at hand*

I want to have Python in my computer but I don’t know how to install it. What to do?

Don’t be shy to ask around.

Take help from friends, teachers, or research scholars in your institute.

Mathematics and Numerical Methods

Equations	Root-finding (Newton-Raphson, Simplex, etc.)
Functions	Optimization (Newton-Raphson, Simplex, etc.)
Function approximation	Least-squares regression (Gaussian elimination, LU decomposition, etc.), interpolation
Eigenvalue problems	Similarity transformation (Jacobi method), Iterative method (Power method, Lanczos, etc.)
Differentiation/Integration	Finite-derivatives, Trapezoidal method, Quadratures, etc.
Differential equations	Euler method, Runge-Kutta, etc.

A warm up problem

1.0 atm of nitrosyl chloride is introduced into a reaction vessel. The compound dissociates into nitric oxide and chlorine according to the reaction

$$2\mathrm{NOCl(g)} \rightleftharpoons 2\mathrm{NO(g)} + \mathrm{Cl_2(g)}$$

If the equilibrium constant of the reaction is known to be 2.18, find the partial pressure of the gases at equilibrium

Answer

At equilibrium, $P_{\mathrm{NOCl}} = 1 - 2x$, $P_{\mathrm{NO}} = 2x$, and $P_{\mathrm{Cl_2}} = x$.

Equilibrium-constant implies $\dfrac{P_{\mathrm{NO}}^2 P_{\mathrm{Cl_2}}}{P_{\mathrm{NOCl}}^2} = \dfrac{(2x)^2 x}{(1-2x)^2} = K_{\mathrm{eq.}} = 2.18$

The expression can be rearranged as a cubic equation $4x^3 - 8.72x^2 + 8.72x - 2.18 = 0$ which needs to be solved to determine the value of x (from which the partial pressures can be calculated)

Since $1 \geq P_{\mathrm{NOCl}} \geq 0$ we know that $1 \geq 1 - 2x \geq 0$ or $x \geq 1/2$.

Cubic equation

We know how to solve a quadratic equation and find two solutions.

We are taught to rearrange the cubic equation into simple forms like (for example)

$(x-d)(ax^2+bx+c)=0$, in order to find the third solution.

Cubic equation

We know how to solve a quadratic equation and find two solutions.

We are taught to rearrange the cubic equation into simple forms like (for example)

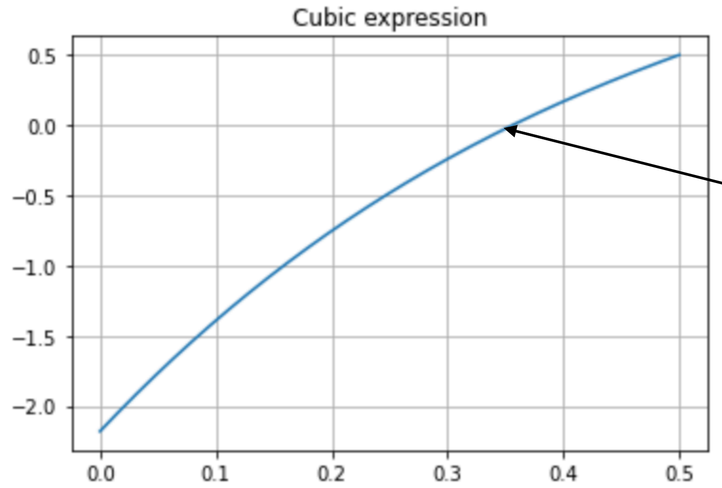
$(x - d)(ax^2 + bx + c) = 0$, in order to find the third solution.

Now, let's see if a computer and Python can help us!

Graphical solution of the cubic equation

In [1]:
```
import numpy as np
import matplotlib.pyplot as plt

#=== x-values
x_min=0.0
x_max=0.5
x_grids=501
x=np.linspace(x_min, x_max, x_grids)

#=== f(x) values
f=4*x**3 - 8.72*x**2 + 8.72*x - 2.18

#=== plot
plt.plot(x,f)
plt.title('Cubic expression')
plt.grid()
plt.show()
```

x-range is fixed between 0 and 1/2 (using our previous knowledge of the problem)



The solution seems to be around x= 0.35

Numerical solution using secant method

In [2]:
```
from scipy import optimize

def f(x):
    f=4*x**3 - 8.72*x**2 + 8.72*x - 2.18
    return f

xguess=0.1
solution=optimize.root_scalar(f, fprime=None, x1=xguess+0.01, method='secant', bracket=None, x0=xguess, \
                      options={'tol':1e-6, 'maxiter':100})
solution
```

Out[2]:
```
      converged: True
           flag: 'converged'
 function_calls: 7
     iterations: 6
           root: 0.3560857818097863
```

- Secant method is a modified version of the Newton-Raphson method.
- Newton-Raphson requires that the derivative of the function is also known.
- In secant method, the derivative is approximated numerically using a finite-step (hence, finite-derivative)
- So, along with x0 (the initial guess for the root), another point x1 must also be specified.
- The derivative for the first iteration is estimated as $f'(x_0) \approx (f(x_1) - f(x_0))/(x_1 - x_0)$

The answer

At equilibrium, $P_{\mathrm{NOCl}} = 1 - 2x$, $P_{\mathrm{NO}} = 2x$, and $P_{\mathrm{Cl_2}} = x$.

In [3]:
```
x=solution.root
print("The partial pressure of NOCl is ",1-2*x)
print("The partial pressure of NO is ",2*x)
print("The partial pressure of Cl2 is ",x)
```

```
The partial pressure of NOCl is  0.28782843638042743
The partial pressure of NO is  0.7121715636195726
The partial pressure of Cl2 is  0.3560857818097863
```

The answer

At equilibrium, $P_{\mathrm{NOCl}} = 1 - 2x$, $P_{\mathrm{NO}} = 2x$, and $P_{\mathrm{Cl_2}} = x$.

```
In [3]: x=solution.root
        print("The partial pressure of NOCl is ",1-2*x)
        print("The partial pressure of NO is ",2*x)
        print("The partial pressure of Cl2 is ",x)

        The partial pressure of NOCl is  0.28782843638042743
        The partial pressure of NO is  0.7121715636195726
        The partial pressure of Cl2 is  0.3560857818097863
```

Are you surprised that the sum of all partial pressures exceed the initial pressure of 1 atm?

Homework

Exercise 1: The following gas-phase reaction occurs at 300 K and has an equilibrium constant $K_{\text{eq.}} = 26$.

$$\text{CH}_4(\text{g}) + \text{H}_2\text{O}(\text{g}) \rightleftharpoons 2\text{CO}(\text{g}) + 3\text{H}_2(\text{g})$$

Initially 1.0 atm of methane and water (vapour) are introduced into the reaction vessel. Find the partial pressure of the species at equilibrium.

Answer: $P_{\text{CO}} = x = 0.615$ atm. From this, you can find the values for other values.

Homework

Exercise 2: The following equation is encountered in the derivation of the Wien's displacement law. Find the solution of this equation graphically and by using the secant method.

$$e^{-x} + \frac{x}{5} = 1$$

Answer: $x = 4.965$

A problem in integration

Debye's theory of molar heat capacity (Debye-T^3 law) of a monoatomic crystal states

$$\overline{C}_V(T) = 9R\left(\frac{T}{\Theta_\mathrm{D}}\right)^3 \int_0^{\Theta_\mathrm{D}/T} \frac{x^4 e^x}{(e^x-1)^2} dx$$

where R is the gas constant and Θ_D is the Debye temperature. Given that for copper, $\Theta_\mathrm{D} = 309\ \mathrm{K}$, find its molar heat capacity at $T = 90\ \mathrm{K}$.

Experimentally measured value is $14.49\,\mathrm{J}\cdot K^{-1}\cdot \mathrm{mol}^{-1}$

G. K. White and S. J. Collocott, *Heat Capacity of Reference Materials: Cu and W*
Journal of Physical and Chemical Reference Data 13, 1251 (1984)
https://doi.org/10.1063/1.555728

Numerical integration using quadrature

In [1]:
```
import numpy as np
from scipy import integrate

R=8.314     # gas constant in J K^-1 mol^-1

Theta_D=309 # Debye Temperature of copper in K

T = 90     # given K at which we want molar heat capacity

def fn_I(x):
    fn_I= x**4 * np.exp(x) / (np.exp(x)-1)**2
    return fn_I

# For fine-tuning the integration settings see the scipy documenation
# https://docs.scipy.org/doc/scipy/reference/generated/scipy.integrate.quadrature.html

Integral=integrate.quadrature(fn_I, 0.0, Theta_D/T)

print("The value of the integral is: ",Integral[0], " with a numerical error of ", Integral[1])

CV=9*R*(T/Theta_D)**3*Integral[0]

print("Heat capacity of Cu at T = 103 K is: ",CV, " J mol^-1 K^-1")
```

lower limit

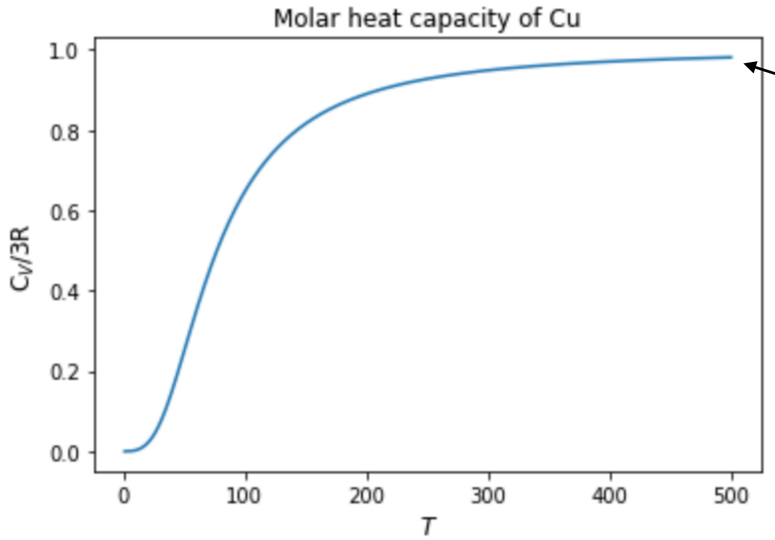
upper limit

```
The value of the integral is:  7.9788851408858035  with a numerical error of  6.324787538147802e-08
Heat capacity of Cu at T = 103 K is:  14.751861725666032  J mol^-1 K^-1
```

Agrees well with the experimentally measured value: $14.49\,\mathrm{J}\cdot\mathrm{mol}^{-1}\cdot K^{-1}$

Bonus: Plotting heat-capacities as a function of T

In [2]:
```
import matplotlib.pyplot as plt

T_i=1
T_f=500
N_T=501
T=np.linspace(T_i, T_f, N_T)

Cv=np.zeros(501)

for i in range(501):
    Integral=integrate.quadrature(fn_I, 0.0, Theta_D/T[i])
    Cv[i]=9*R*(T[i]/Theta_D)**3*Integral[0]

plt.plot(T,Cv/(3*R))
plt.title('Molar heat capacity of Cu', fontsize=12)
plt.xlabel("$T$", fontsize=12)
plt.ylabel("C$_V$/3R", fontsize=12)
plt.show()
```



approaches the Dulong-Petit limit at hight temperature

Homework

Exercise 3: The following integral is encountered in derivation of Stefan-Boltzmann's law. Determine its value using quadrature.

$$I = \int_0^\infty \frac{x^3}{e^x - 1} dx$$

Hint: Try different upper bounds like1,10,100 and see how the integral converges.

Answer: $I = \frac{\pi^4}{15} = 6.494$

Energy of the simplest molecule, $\mathbf{H}_2^+$

One of the most interesting and useful applications of the variational method in quantum chemistry is to derive the total energy of the one-electron molecule $\mathbf{H}_2^+$ (Hydrogen molecule cation).

In this exercise, the H atoms are placed symmetrically at

$$\vec{r}_A = -\vec{R}/2 \text{ and } \vec{r}_B = +\vec{R}/2.$$

Each H-atom contains a $1s$ orbital, described by the wave functions

$$\phi_A(\vec{r}) = \sqrt{\frac{\zeta^3}{\pi}} \exp(-\zeta |\vec{r} - \vec{r}_A|) \text{ and } \phi_B(\vec{r}) = \sqrt{\frac{\zeta^3}{\pi}} \exp(-\zeta |\vec{r} - \vec{r}_B|)$$

See: Quantum Chemistry, Donald A. McQuarrie (chapter 9)

Energy of the simplest molecule, H_2^+

The wave function for the bonding molecular orbital (which is the ground state of the H_2^+ molecule) turns out to be

$$\psi_{\text{bonding}} = \frac{1}{\sqrt{2}}\left[\phi_A(\vec{r}) + \phi_B(\vec{r})\right]$$

while the corresponding total energy of the molecule is the energy expectation value is a function of the exponent ζ and the inter-atomic distance $R = |\,\overrightarrow{r_A} - \overrightarrow{r_B}\,|$

$$E = \langle\psi|\,\hat{H}\,|\psi\rangle = f(\zeta, R)$$

According to variational principle, the ground state energy is given as the minimum energy (i.e. most negative energy) that can be obtained by varying the parameters ζ and R.

Experimental bond length is $R = 1.057\text{Å}$ and the dissociation energy is $D_e = 2.79$ eV

Energy of the simplest molecule, $\mathbf{H}_2^+$

The final expression is given in textbooks as

$$E(\zeta, R) = \frac{J(\zeta, R) + K(\zeta, R)}{1 + S(\zeta, R)} + \frac{1}{R}$$

where the terms $J(\zeta, R)$, $K(\zeta, R)$, and $S(\zeta, R)$ are Coulomb, exchange and overlap integrals.

$S(\zeta, R) = \left(1 + \zeta R + \frac{1}{3}\zeta^2 R^2\right) \exp\left(-\zeta R\right)$, $J(\zeta, R) = \frac{1}{2}\zeta^2 - \zeta - \frac{1}{R} + \left(\frac{1}{R} + \zeta\right) \exp\left(-2\zeta R\right)$, and

$K(\zeta, R) = -\frac{1}{2}\zeta^2 S(\zeta, R) - \zeta(1 + \zeta R)(2 - \zeta)\exp\left(-\zeta R\right)$.

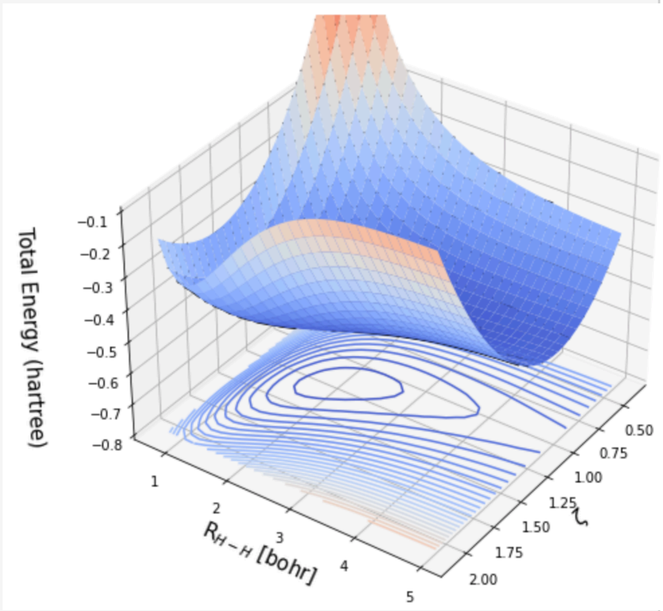
The term $\frac{1}{R}$ accounts for internuclear repulsion.

See: Quantum Chemistry, Donald A. McQuarrie (chapter 9)

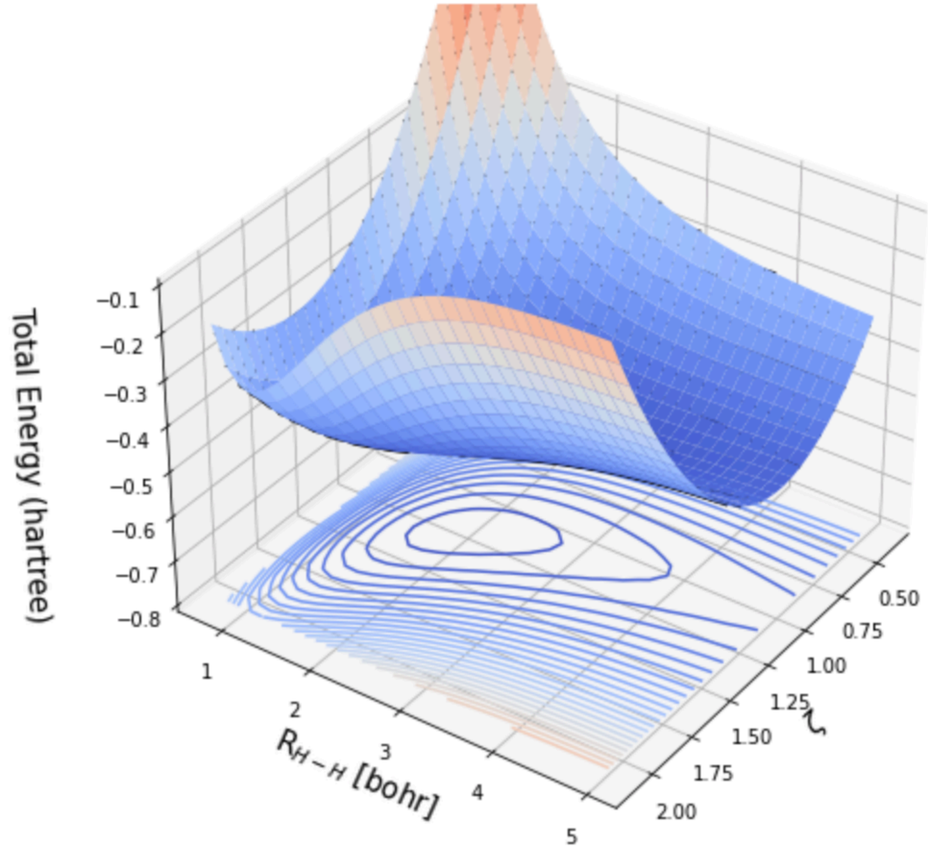
Finding the minimum energy value graphically

In [1]:

```
import numpy as np
from mpl_toolkits.mplot3d import axes3d
import matplotlib.pyplot as plt

def E(x,y):
    S=(1 + x*y + (1/3)*(x*y)**2) * np.exp(-x*y)
    J=(1/2)*x**2 - x -1/y + (1/y+x) * np.exp(-2*x*y)
    K=-(1/2) * S*x**2 - x * (1+x*y) * (2-x) * np.exp(-x*y)
    E=( J + K )/(1 + S) + 1/y
    return E

plt.rcParams["figure.figsize"] = (10,8)
fig = plt.figure()
ax = fig.add_subplot(111, projection="3d")
X, Y = np.mgrid[0.4:2.1:30j, 0.8:5:30j]

Z = E(X,Y)

ax.plot_surface(X, Y, Z, cmap="coolwarm", lw=1, rstride=1, cstride=1)
ax.contour(X, Y, Z, 40, cmap="coolwarm", linestyles="solid", offset=-0.8)
ax.contour(X, Y, Z, 40, colors="k", linestyles="solid")

ax.set_xlabel('$\zeta$',fontsize=15)
ax.set_ylabel('R$_{H-H}$ [bohr]',fontsize=15)
ax.set_zlabel('Total Energy (hartree)',fontsize=15,labelpad=20)

ax.set_zlim3d(-0.8,-0.1)

ax.view_init(35, 35)

plt.show()
```



Finding the minimum energy of H_2^+ graphically



In [2]:
```
x_guess = 1.1 # zeta
y_guess = 2.0 # R in bohr
print('Eye ball estimation of the minimum energy is: ', E(x_guess,y_guess), ' hartree')
```

```
Eye ball estimation of the minimum energy is:  -0.5754658938005228  hartree
```

Finding the minimum energy by optimization

In [3]:
```
from scipy import optimize

def E(x):
    zeta=x[0]
    R=x[1]
    S=(1 + zeta*R + (1/3)*(zeta*R)**2) * np.exp(-zeta*R)
    J=(1/2)*zeta**2 - zeta -1/R + (1/R+zeta) * np.exp(-2*zeta*R)
    K=-(1/2) * S*zeta**2 - zeta * (1+zeta*R) * (2-zeta) * np.exp(-zeta*R)
    E=( J + K )/(1 + S) + 1/R
    return E

x=[1,1]
solution=optimize.minimize(E,x,method='L-BFGS-B')
```

Generalized secant method for higher dimensional problems

In [4]:
```
print('Best exponent is $\zeta$: ', solution.x[0])
```

```
Best exponent is $\zeta$:  1.23802937000445
```

In [5]:
```
print('Best bond length $R$: ', solution.x[1], ' bohr = ',solution.x[1]*0.529177208, ' Angstrom')
```

```
Best bond length $R$:  2.0032944446083936  bohr =  1.0600977609997804  Angstrom
```

In [6]:
```
print('Best estimation of energy is: ', solution.fun, ' hartree = ', solution.fun*27.21138386937795, ' eV' )
```

```
Best estimation of energy is:  -0.5865065021564653  hartree =  -15.959653572065724  eV
```

In [7]:
```
print('Dissociation energy (atoms - molecules) is ',(-0.5-solution.fun)*27.21138386937795, ' eV')
```

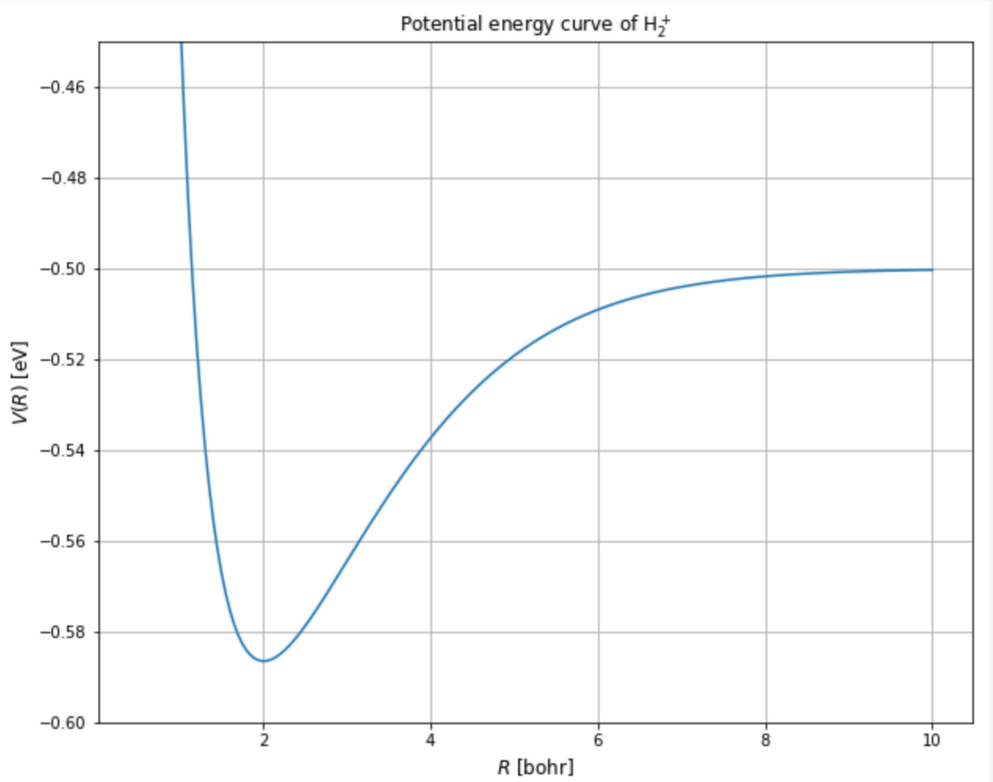
```
Dissociation energy (atoms - molecules) is  2.353961637376748  eV
```

These value agree well with the experimental values: $R = 1.057\text{Å}$ and $D_e = 2.79$ eV

Potential energy curve of H_2^+

For different values of R, one can minimize E as a function ζ to obtain a potential energy curve

In [8]:
```
def E(x):
    zeta=x
    S=(1 + zeta*R + (1/3)*(zeta*R)**2) * np.exp(-zeta*R)
    J=(1/2)*zeta**2 - zeta -1/R + (1/R+zeta) * np.exp(-2*zeta*R)
    K=-(1/2) * S*zeta**2 - zeta * (1+zeta*R) * (2-zeta) * np.exp(-zeta*R)
    E=( J + K )/(1 + S) + 1/R
    return E

V=np.zeros(191)
r=np.zeros(191)

for i in range(191):
    R=0.5+i*0.05
    x = 1.0
    solution=optimize.minimize(E,R,method='L-BFGS-B')
    r[i]=R
    V[i]=solution.fun

plt.plot(r,V)
plt.title('Potential energy curve of H$_2^+$')
plt.savefig('pes.png')
plt.ylim(-0.60,-0.45)
plt.grid()
plt.xlabel("$R$ [bohr]", fontsize=12)
plt.ylabel("$V(R)$ [eV]", fontsize=12)
plt.show()
```



How to determine the force constant?

Recall Hooke's law and you will find out that force constant is the second derivative of potential energy calculated at the minimum energy bond length $F = -kx \implies -\frac{dV}{dx} = -kx \implies \frac{d^2V}{dx^2} = k$

One can follow 2 numerical approaches. We will try the first approach today.

1. Determine k using finite-derivatives of V around the minimum energy bond length

$$k \approx \frac{V(R_e + \Delta x) + V(R_e - \Delta x) - 2V(R_e)}{\Delta x^2}$$

2. Model the potential energy around the minimum energy bond length as a Taylor polynomial using least-squares regression (also known as least-squares fitting)

$$V(R) = V(R_e) + \frac{dV}{dR}(R - R_e) + \frac{1}{2!}\frac{d^2V}{dR^2}(R - R_e)^2 + \ldots$$

Force constant by finite-derivative

In [1]:

```
import numpy as np
from scipy import optimize

def E(x):
    zeta=x
    S=(1 + zeta*R + (1/3)*(zeta*R)**2) * np.exp(-zeta*R)
    J=(1/2)*zeta**2 - zeta -1/R + (1/R+zeta) * np.exp(-2*zeta*R)
    K=-(1/2) * S*zeta**2 - zeta * (1+zeta*R) * (2-zeta) * np.exp(-zeta*R)
    E=( J + K )/(1 + S) + 1/R
    return E

R0=2.0032944446083936
dR = 0.1   #=== dR, step size of 0.1

R=R0
solution=optimize.minimize(E,1.0,method='L-BFGS-B',options={'ftol': 1e-15})
V0=solution.fun

R=R+dR
solution=optimize.minimize(E,1.0,method='L-BFGS-B',options={'ftol': 1e-15})
V_plus=solution.fun

R=R-dR
solution=optimize.minimize(E,1.0,method='L-BFGS-B',options={'ftol': 1e-15})
V_minus=solution.fun

k = (V_plus+V_minus-2*V0)/(dR**2)

print("Force constant of hydrogen molecule cation is: ", k, " hartree/bohr^2")
```

```
Force constant of hydrogen molecule cation is:  [0.04297412]  hartree/bohr^2
```

Fine-tuning the step size

In [2]:
```
for i in range(6):

    dR=1/10**i

    R=R+dR
    solution=optimize.minimize(E,1.0,method='L-BFGS-B',options={'ftol': 1e-15})
    V_plus=solution.fun

    R=R-dR
    solution=optimize.minimize(E,1.0,method='L-BFGS-B',options={'ftol': 1e-15})
    V_minus=solution.fun

    k = (V_plus+V_minus-2*V0)/(dR**2)

    print("For dR=", dR, " force constant is", k, " hartree/bohr^2")
```

```
For dR= 1.0  force constant is [0.0221568]  hartree/bohr^2
For dR= 0.1  force constant is [0.04297412]  hartree/bohr^2
For dR= 0.01  force constant is [0.04641588]  hartree/bohr^2
For dR= 0.001  force constant is [0.04668428]  hartree/bohr^2
For dR= 0.0001  force constant is [0.04574696]  hartree/bohr^2
For dR= 1e-05  force constant is [0.03599565]  hartree/bohr^2
```

Between successive step sizes, the difference is least for dR=0.01 and dR=0.001. For smaller steps, the estimation diverges. So it is likely that the best estimate is

$k = 0.04668$ hartree/bohr^2

Harmonic wavenumber from the force constant

Applying the simple spring-model of diatomic molecule, we have $\nu = \frac{1}{2\pi}\sqrt{\frac{k}{\mu}} \implies \bar{\nu} = \frac{1}{2\pi c}\sqrt{\frac{k}{\mu}}$

In [3]:
```
# Factors for unit conversion
hartree2j = 4.359743941424844e-18   # hartree to J
bohr2m    = 5.291772083164631e-11   # bohr to m
amu2kg    = 1.660538782000000e-27   # amu to kg

# Force constant
k = 0.04668                         # in hartree/bohr^2
k = k * hartree2j/(bohr2m**2)       # in J/m^2

# Speed of light
c = 2.99792458 * 10**8              # m/s
c = c*100                           # in cm/s

# Mass, reduced-mass
mass = 1.008                        # in amu
mu=mass/2 * amu2kg                  # in kg, mu=m1*m2/(m1+m2)

# Wave number
nubar=(1/(2*np.pi*c))*np.sqrt(k/mu) # in cm^-1

print('Harmonic wavenumber of H_2^+ molecule is:', nubar, ' cm^-1')
```

```
Harmonic wavenumber of H_2^+ molecule is: 1564.423867968304  cm^-1
```

The experimental value from IR/Raman vibrational spectroscopy is 1148.1 cm^-1

Homework

Exercise 4: What are the various factors responsible for the deviation of the wavenumber of the molecule estimated using harmonic force constant (based on the variational energy model) from the experimental wavenumber.

Can you think of a simple reason to explain why the estimated wavenumber is larger than the actual, experimental value?

Hueckel Molecular Orbitals for pi-electrons

The Hueckel molecular orbital theory is very popular for determining the qualitative shapes and energies of pi (bonding and anti-bonding) molecular orbitals. In this method, the electronic Hamiltonian of the molecule is defined using bonding connectivities.

- The diagonal elements of the Hamiltonian matrix ($H_{i,i}$) is set to a constant α, corresponding to atomic energies. Conventionally α is set to zero.
- The off-diagonal elements of the Hamiltonian matrix ($H_{i,j}$) is set to a negative number β. Negative value indicates that the bonding interaction between atoms is stabilising or attractive. Conventionally β is set to -1.
- IMPORTANT: $H_{i,j} = \beta$ only when *i* and *j* denote atoms that are connected (nearest neighbours). Otherwise $H_{i,j} = 0$.
- The eigenvalues of the Hamiltonian matrix denote the energies of the pi-MOs
- The eigenvectors contain information about the phase of the MO wave function and also indicate which atoms contribute to a particular MO.

Hueckel MO Hamiltonian matrix

for ethylene

$$\begin{pmatrix} 0 & -1 \\ -1 & 0 \end{pmatrix}$$

for 1,3-butadiene

$$\begin{pmatrix} 0 & -1 & 0 & 0 \\ -1 & 0 & -1 & 0 \\ 0 & -1 & 0 & -1 \\ 0 & 0 & -1 & 0 \end{pmatrix}$$

for benzene

$$\begin{pmatrix} 0 & -1 & 0 & 0 & 0 & -1 \\ -1 & 0 & -1 & 0 & 0 & 0 \\ 0 & -1 & 0 & -1 & 0 & 0 \\ 0 & 0 & -1 & 0 & -1 & 0 \\ 0 & 0 & 0 & -1 & 0 & -1 \\ -1 & 0 & 0 & 0 & -1 & 0 \end{pmatrix}$$

Eigenvalues and Eigenvectors

In [1]:
```
import numpy as np
from numpy import linalg as npla

def eig(A):
    eigenValues, eigenVectors = npla.eig(A)
    idx = np.argsort(eigenValues)
    eigenValues = eigenValues[idx]
    eigenVectors = eigenVectors[:,idx]
    return (eigenValues, eigenVectors)

# Ethylene
H=np.zeros([2,2])

H[0][1]=-1
H[1][0]=-1

print(H)
```

```
[[ 0. -1.]
 [-1.  0.]]
```

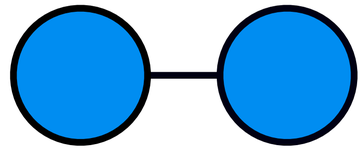
Energy of bonding MO (negative means stabilising)

Energy of anti-bonding MO

In [2]:
```
E,V=eig(H)
```

In [3]:
```
print(E)
```

```
[-1.  1.]
```

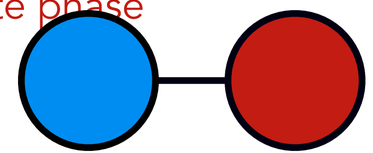


Eigenvector of bonding MO, both coefficients in same phase

In [4]:
```
print(V)
```

```
[[ 0.70710678  0.70710678]
 [ 0.70710678 -0.70710678]]
```

Eigenvector of anti-bonding MO, both coefficients in opposite phase



1,3-butadiene

In [5]:
```
# 1,3-Butadiene
H=np.zeros([4,4])

for i in range(4):
    for j in range(4):
        if np.abs(i-j)==1:
            H[i][j]=-1

print(H)
```

```
[[ 0. -1.  0.  0.]
 [-1.  0. -1.  0.]
 [ 0. -1.  0. -1.]
 [ 0.  0. -1.  0.]]
```

In [6]:
```
E,V=eig(H)
```

In [7]:
```
print(E)
```

```
[-1.61803399 -0.61803399  0.61803399  1.61803399]
```

In [8]:
```
print(V)
```

```
[[ 0.37174803 -0.60150096  0.60150096 -0.37174803]
 [ 0.60150096 -0.37174803 -0.37174803  0.60150096]
 [ 0.60150096  0.37174803 -0.37174803 -0.60150096]
 [ 0.37174803  0.60150096  0.60150096  0.37174803]]
```

Least energy MO has no nodes

Next MO has 1 node

has 2 nodes

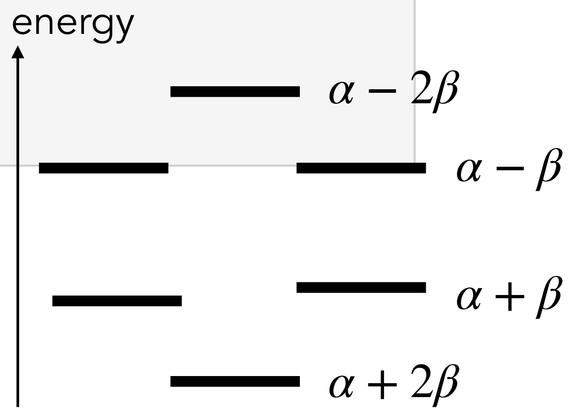
has 3 nodes

Benzene

```
In [9]: # Benzene
H=np.zeros([6,6])

for i in range(6):
    for j in range(6):
        if np.abs(i-j)==1:
            H[i][j]=-1
H[0][5]=-1
H[5][0]=-1
print(H)
```

```
[[ 0. -1.  0.  0.  0. -1.]
 [-1.  0. -1.  0.  0.  0.]
 [ 0. -1.  0. -1.  0.  0.]
 [ 0.  0. -1.  0. -1.  0.]
 [ 0.  0.  0. -1.  0. -1.]
 [-1.  0.  0.  0. -1.  0.]]
```

Doubly degenerate HOMO and LUMO




```
In [10]: E,V=eig(H)
```

```
In [11]: print(E)
```

```
[-2. -1. -1.  1.  1.  2.]
```

```
In [12]: print(V)
```

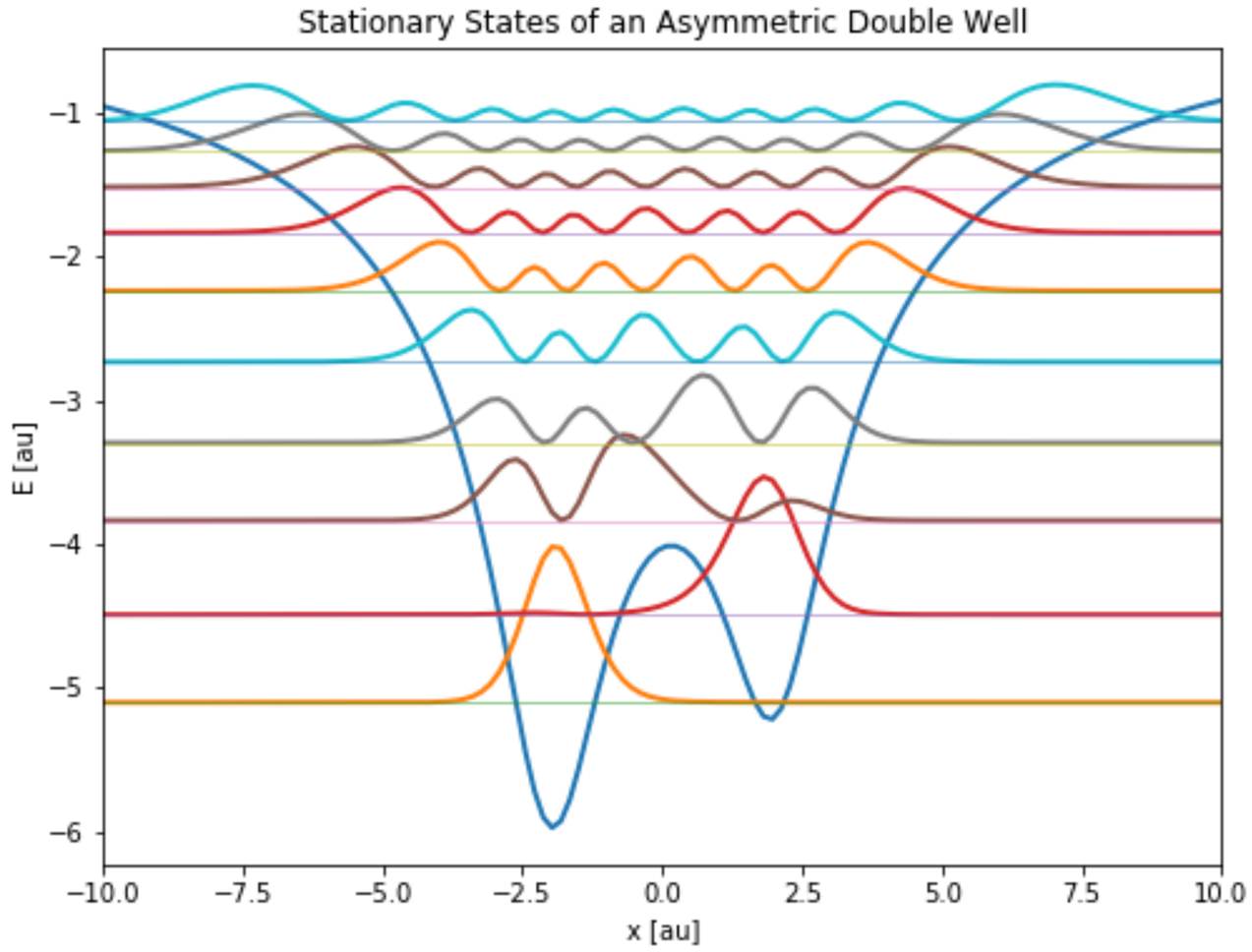
```
[[ 0.40824829  0.57245374  0.07503368  0.57735027 -0.09791921  0.40824829]
 [ 0.40824829  0.22124579  0.53327632 -0.28867513 -0.4437968  -0.40824829]
 [ 0.40824829 -0.35120794  0.45824264 -0.28867513  0.54171601  0.40824829]
 [ 0.40824829 -0.57245374 -0.07503368  0.57735027 -0.09791921 -0.40824829]
 [ 0.40824829 -0.22124579 -0.53327632 -0.28867513 -0.4437968   0.40824829]
 [ 0.40824829  0.35120794 -0.45824264 -0.28867513  0.54171601 -0.40824829]]
```

Analyse the nodal structure of the MOs

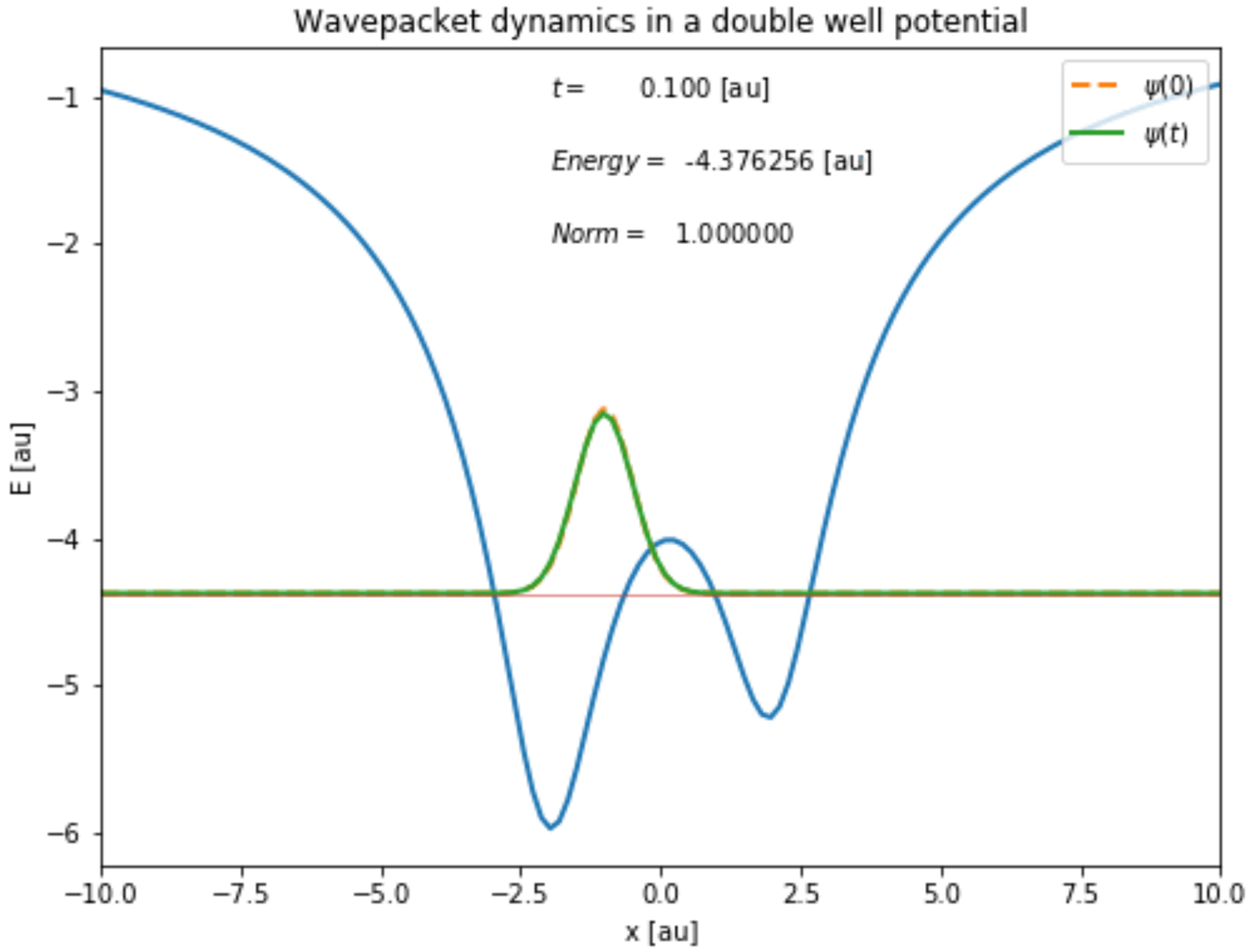
A quick glance at the endless possibilities

Ideas for mini-projects

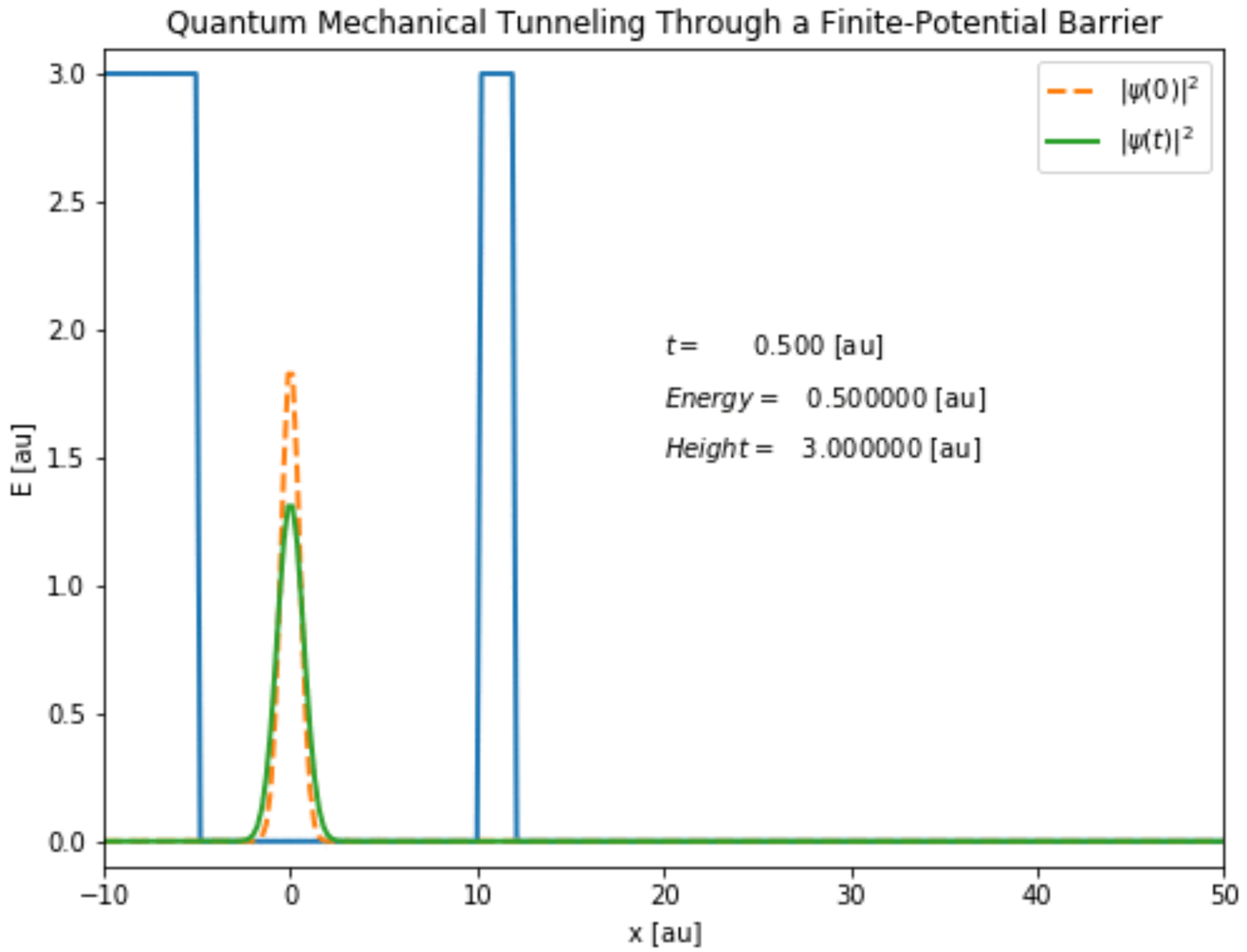
Eigenvalues and Eigenfunctions of QM problems



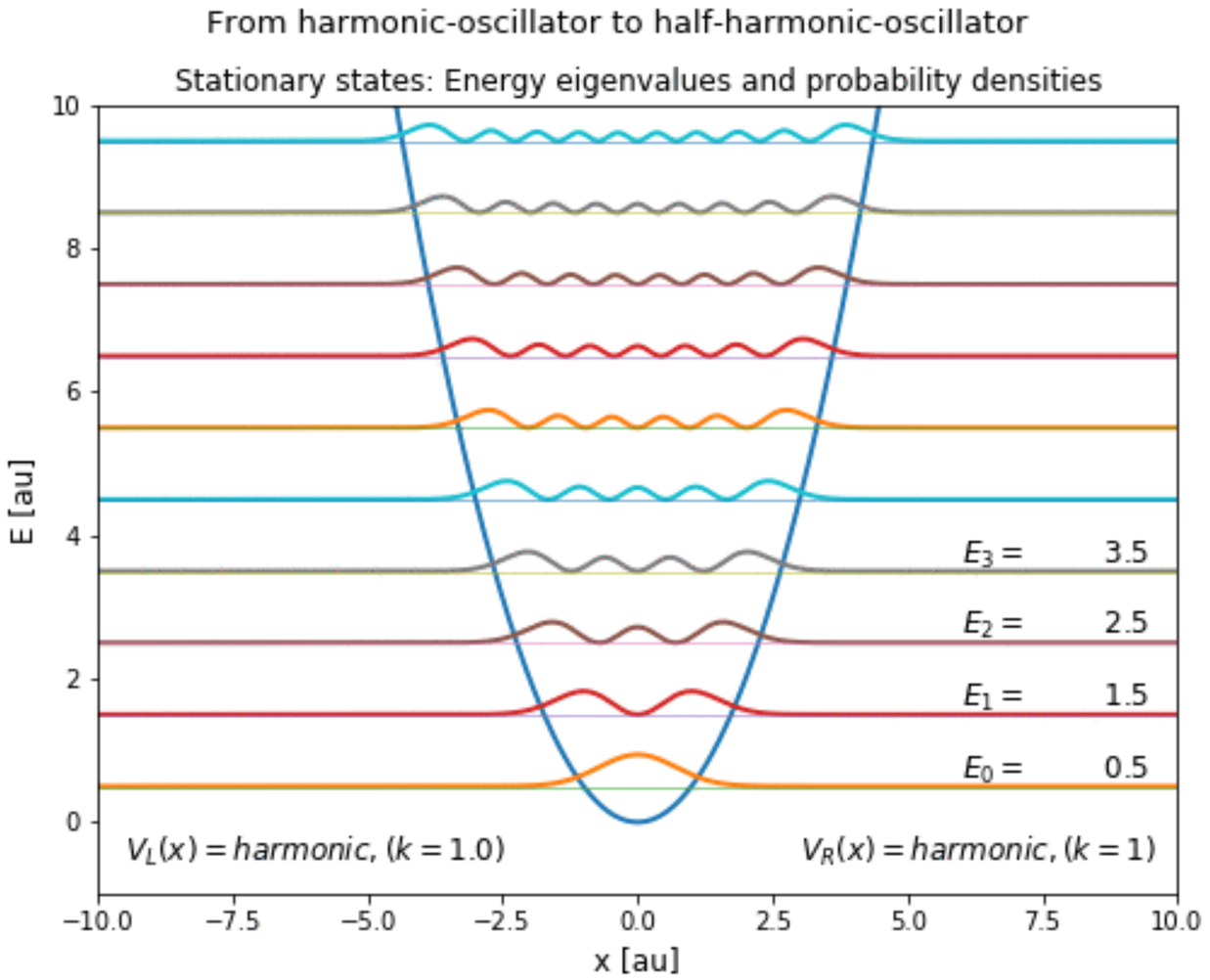
Wavepacket dynamics to understand Tunneling



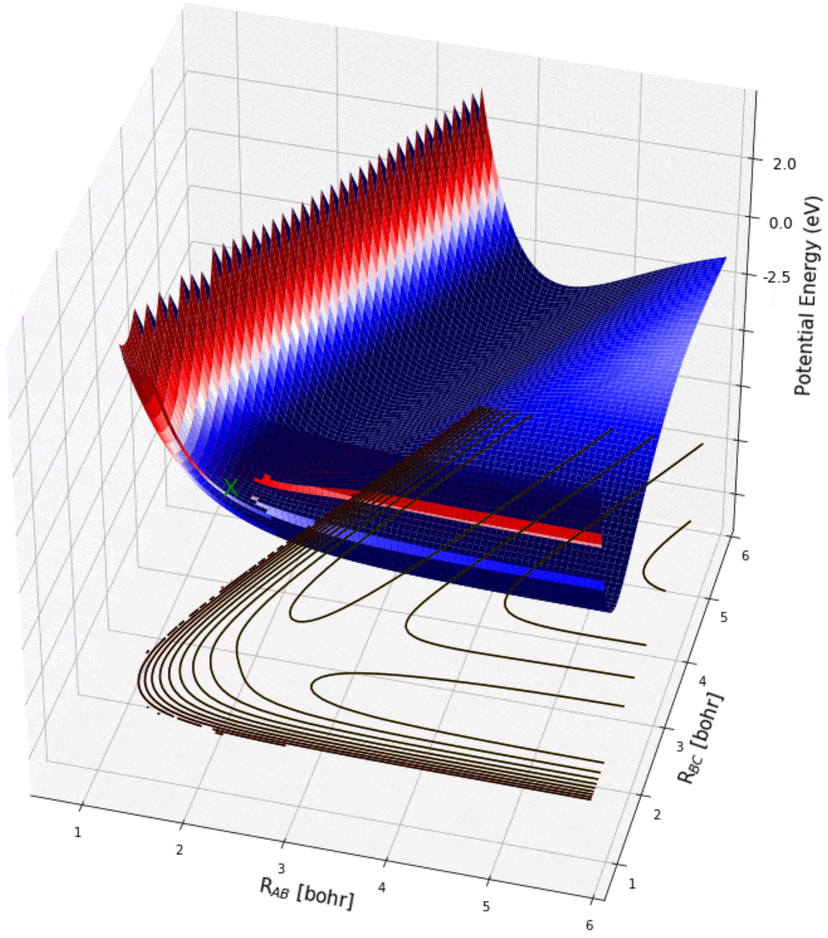
More Tunneling problems



From harmonic to half-harmonic oscillator



Quasi-classical trajectory for H-exchange reaction



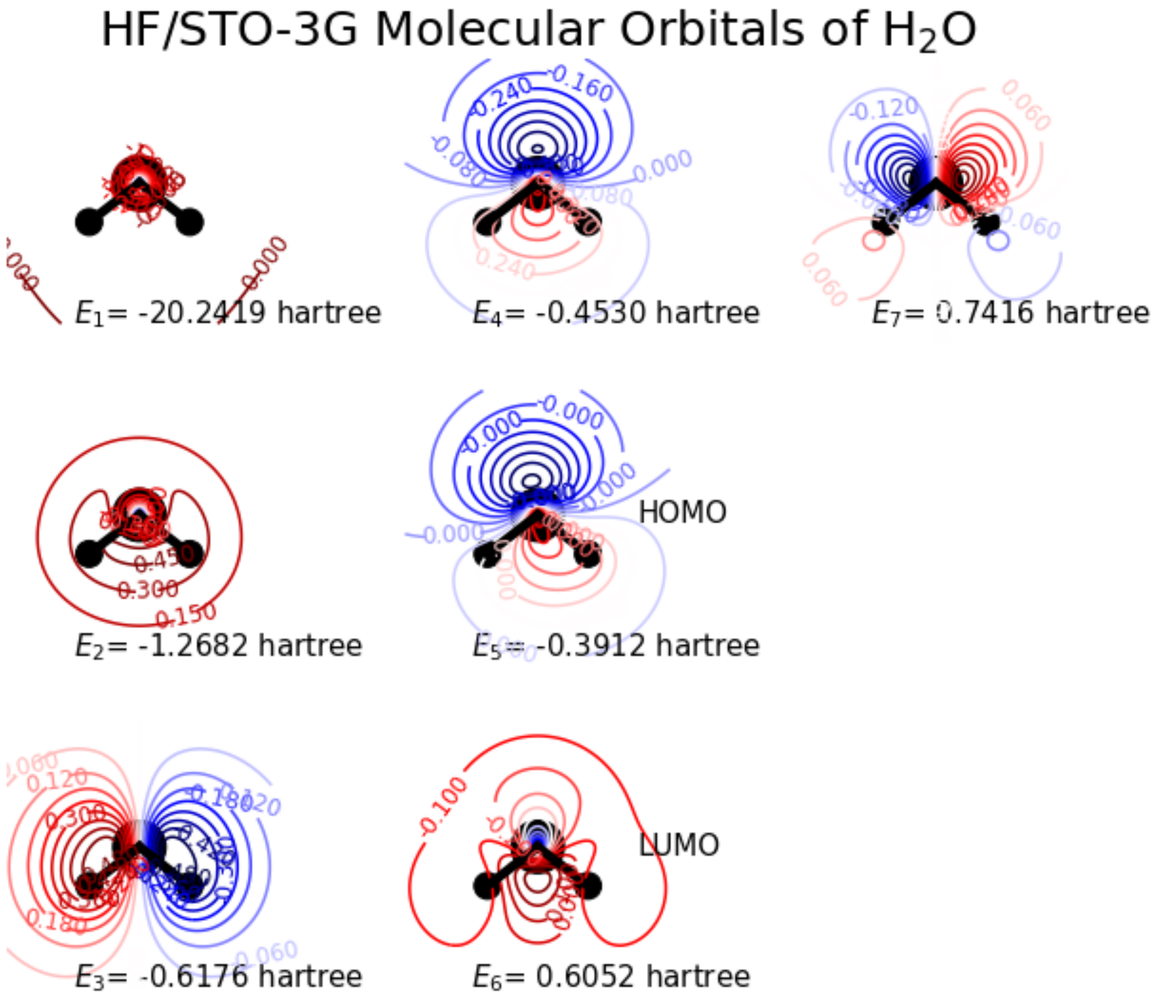
H_A H_B

0.90

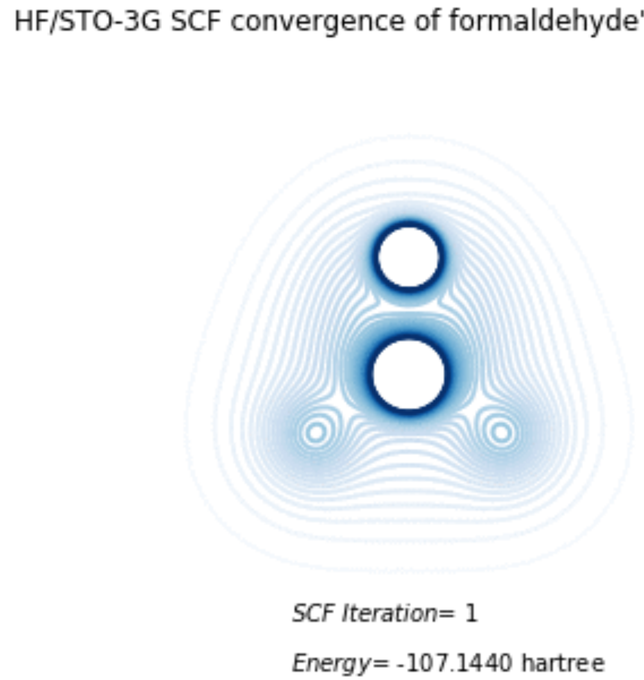
5.55

H_C

Molecular orbitals of water molecule



Hartree-Fock calculation of formaldehyde



Thank you for your attention!

ramakrishnan@tifrh.res.in