Noen tips til den numeriske øvingen

Definere potensialet $V(x)$ som et numpy array

Hvis et heltall `N` og et potensial `V0` er definert, kan et harmonisk potensial med "kontakter" med konstant potensial `V0` for eksempel lages slik:

```
N=100
V = [V0]*4*N + [V0*((n-N)/(N*1.0))**2 for n in range(2*N+1)] + [V0]*4*N
V = np.asarray(V)
```

Lage stasjonære tilstander og tilhørende energiegenverdier

Hvis vi har `Ntot` punkter langs x-aksen, får vi `Ntot` ortonormerte egenfunksjoner og energiegenverdier ved å bruke numpyfunksjonen `linalg.eigh`, på samme måte som i eksempelprogrammene som ligger tilgjengelig:

```
d = [v + hbar**2/(m*dx**2) for v in V]
e = - hbar**2/(2*m*dx**2)
Ntot=len(V)
H = [[0]*(Ntot) for n in range(Ntot)]
for i in range(Ntot):
    for j in range(Ntot):
        if i==j:
            H[i][j]=d[i]
        if abs(i-j)==1:
            H[i][j]=e

H = np.asarray(H)
energy,psi_matrix = np.linalg.eigh(H)
```

Siden matrisen `H` er tridiagonal, finnes det muligens mer effektive funksjoner enn `eigh`.

Starttilstanden

En gaussformet starttilstand med senterposisjon `x0` og romlig utstrekning `sigma` kan for eksempel lages omtrent slik (se Oppgave 7 i øvingene):

```
dx = 1.0E-10
x = np.asarray([dx*n for n in range(Ntot)])
x0 = 50*N*dx
sigma = 10*N*dx
normfactor = (2*np.pi*sigma**2)**(-0.25)
gaussinit = np.exp(-(x-x0)**2/(4*sigma**2))
planewavefactor = np.exp(1j*k0*x)
Psi0 = normfactor*gaussinit*planewavefactor
```

Planbølgefaktoren $\exp(ik_0x)$ sørger for at bølgepakken har middelimpuls $p_0 = \hbar k_0$ og middelhastighet $v_0 = p_0/m = \hbar k_0/m$.

Bølgepakken som lineærkombinasjon av stasjonære tilstander

Nå kan bølgepakken $\Psi(x,t)$ utvikles i de `Ntot` stasjonære tilstandene $\psi_n(x)$:

$$\Psi(x,t) = \sum_{n=0}^{N_{\text{tot}}-1} c_n \psi_n(x) \exp(-iE_n t/\hbar).$$

Multiplikasjon på begge sider med $\psi_j^*(x)$ og integrasjon over x gir nå, med $t = 0$ (siden funksjonene $\psi_j(x)$ er ortonormerte),

$$c_j = \int \psi_j^*(x)\, \Psi(x,0)\, dx.$$

Dette integralet kan regnes ut som en sum i programmet:

```
psi_matrix_complex = psi_matrix*(1.0 + 0.0j)
c = np.zeros(Ntot, dtype = np.complex128)
for n in range(Ntot):
    c[n] = np.vdot(psi_matrix_complex[:,n],Psi0)
```

Nå er alt som inngår i $\Psi(x,t)$ kjent.

Animasjon av bølgepakke

Oppskriften følger i samme spor som i utlagte programmer:

```
#Forbereder figur, akser, og plotteelementet som skal animeres:
fig = plt.figure('Wave packet animation' , figsize=(16,8))
ymax = 1.0E8   #Justeres etter behov, eller beregnes
ax = plt.axes(xlim=(0, Ntot*dx), ylim=(0, ymax))
line, = ax.plot([], [], lw=1)

#Initialisering; plotter bakgrunnen:
def init():
    line.set_data([], [])
    return line,

#Setter tidssteget, justeres etter behov, eller beregnes
tidssteg = 3.0E-15
#Animasjonsfunksjon; kalles sekvensielt:
def animate(i):
    t = i*tidssteg
    #Beregner Psi(x,t)
    Psi_t = np.zeros(Ntot,dtype=np.complex128)
    for n in range(Ntot):
        Psi_t = Psi_t + c[n]*psi_matrix_complex[:,n]*np.exp(-1j*energy[n]*t/hbar)

    rho_t = np.abs(Psi_t)**2
    line.set_data(x, rho_t)
    return line,

plt.plot(x,V*ymax/Vmax)
plt.xlabel('$x$ (m)',fontsize=20)
#Kaller animatoren.
#frames=antall bilder, dvs maxverdi for i;
#interval=varighet av hvert bilde i animasjonen, i millisekunder
anim = animation.FuncAnimation(fig, animate, init_func=init, repeat=False,
```

```
                                frames=200, interval=1, blit=True)

plt.show()
```

Beregning av standardavvik

Hvis `rho_t` er en tabell med sannsynlighetstetthet (ved et eller annet tidspunkt) og `x` er en tabell med posisjonsverdier, kan standardavviket i `x` f eks beregnes slik:

```
x2mean[i] = dx*np.dot(x**2,rho_t)
xmean2[i] = dx*dx*np.dot(x,rho_t)**2
deltax = np.sqrt(x2mean-xmean2)
```