2006 Summer School on Computational Materials Science

Lecture Notes: Ab Initio Molecular Dynamics Simulation Methods in Chemistry

Victor S. Batista*

Yale University, Department of Chemistry, P.O.Box 208107, New Haven, Connecticut 06520-8107, U.S.A.

I Solutions to Problems

Problem 1:

In order to visualize the output of this program, cut the source code attached below, save it in a file named Problem1.f, compile it by typing

```
f77 Problem1.f -o Problem1
```

run it by typing

```
./Problem1
```

Visualize the output as follows: type

```
gnuplot
```

then type

```
plot ``arch.0000''
```

That will show the representation of the Gaussian state, introduced in Eq. (6) in terms of an array of numbers associated with a grid in coordinate space. To exit, type

```
quit
```

*E-mail: victor.batista@yale.edu

```
      PROGRAM Problem_1
      call Initialize()
      CALL SAVEWF(0)
      END
ccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccc
      SUBROUTINE Initialize()
c
c     Wave Packet Initialization: Gaussian centered at xk, with momentum pk
c
      IMPLICIT NONE
      INTEGER nptx,npts,kk
      COMPLEX chi,EYE
      REAL omega,xmin,xmax,dx,pi,mass,xk,pk,x,alpha
      PARAMETER(npts=10,nptx=2**npts)
      COMMON / wfunc/ chi(nptx)
      common /xy/  xmin,xmax
      common /packet/mass,xk,pk
      xmin=-20.
      xmax=20.
      EYE=(0.0,1.0)
      pi= acos(-1.0)
      omega=1.
      dx=(xmax-xmin)/real(nptx)
      pk=0.0
      xk=0.0
      mass=1.0
      alpha=mass*omega
      do kk=1,nptx
         x=xmin+kk*dx
         chi(kk)=((alpha/pi)**0.25)
     1        *exp(-alpha/2.*(x-xk)**2+EYE*pk*(x-xk))
      end do
      RETURN
      END
ccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccc
      SUBROUTINE SAVEWF(j)
c
c     Save Wave-packet in coordinate space
c
      IMPLICIT NONE
      INTEGER nptx,npts,kk,j
      COMPLEX chi,EYE
      REAL RV,omega,xmin,xmax,dx,pi,mass,xk,pk,x,alpha,Vpot,RKE
      character*9 B
      PARAMETER(npts=10,nptx=2**npts)
      COMMON / wfunc/ chi(nptx)
      common /xy/  xmin,xmax
      common /packet/mass,xk,pk
      write(B, '(A,i4.4)') 'arch.', j
      OPEN(1,FILE=B)
      dx=(xmax-xmin)/real(nptx)
      do kk=1,nptx
         x=xmin+kk*dx
         WRITE(1,22) x,chi(kk)
      end do
      CLOSE(1)
 22   FORMAT(6(e13.6,2x))
      RETURN
      END
ccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccc
```

Problem 2:

In order to visualize the output of this program, cut the source code attached below, save it in a file named Problem2.f, compile it by typing

```
f77 Problem2.f -o Problem2
```

run it by typing

```
./Problem2
```

Visualize the output as follows: type

```
gnuplot
```

then type

```
plot ``nume.0000''
```

That will show the representation of the amplitude of the Fourier transform of the Gaussian state, introduced in Eq. (6), in terms of an array of numbers associated with a grid in momentum space. In order to visualize the analytic results on top of the numerical values type

```
replot ``anal.0000''
```

In order to visualize the numerically computed phases as a function of p type

```
plot ``nume.0000 u 1:3''
```

and to visualize the analytic results on top of the numerical values type

```
replot ``anal.0000''
```

To exit, type

```
quit
```

```
      PROGRAM Problem2
      call Initialize()
      CALL SAVEFT()
      END
ccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccc
      SUBROUTINE Initialize()
c
c     Wave Packet Initialization: Gaussian centered at xk, with momentum pk
c
      IMPLICIT NONE
      INTEGER nptx,npts,kk
      COMPLEX chi,EYE
      REAL omega,xmin,xmax,dx,pi,rmass,xk,pk,x,alpha
      PARAMETER(npts=10,nptx=2**npts)
      COMMON / wfunc/ chi(nptx)
      common /xy/  xmin,xmax
      common /packet/rmass,xk,pk
      xmin=-20.
      xmax=20.
      EYE=(0.0,1.0)
      pi= acos(-1.0)
      omega=1.
      dx=(xmax-xmin)/real(nptx)
      pk=0.0
      xk=5.0
      rmass=1.0
      alpha=rmass*omega
      do kk=1,nptx
         x=xmin+kk*dx
         chi(kk)=((alpha/pi)**0.25)
     1        *exp(-alpha/2.*(x-xk)**2+EYE*pk*(x-xk))
      end do
      RETURN
      END
ccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccc
      subroutine SAVEFT()
c
c     Save wave-packet in momentum space
c
      IMPLICIT NONE
      INTEGER nptx,kx,nx,npts,j
      REAL theta,wm,p,xmin,xmax,rmass,xk,pi,alenx,pk,rm,re,ri
      COMPLEX eye,chi,Psip
      character*9 B1,B2
      parameter(npts=10,nptx=2**npts)
      common /xy/  xmin,xmax
      common /packet/  rmass,xk,pk
      COMMON / wfunc/ chi(nptx)
      j=0
      write(B1, '(A,i4.4)') 'anal.', j
      OPEN(1,FILE=B1)
      write(B2, '(A,i4.4)') 'nume.', j
      OPEN(2,FILE=B2)
      CALL fourn(chi,nptx,1,-1)
      pi = acos(-1.0)
      alenx=xmax-xmin
      do kx=1,nptx
         if(kx.le.(nptx/2+1)) then
            nx=kx-1
         else
            nx=kx-1-nptx
```

```
         end if
         p=0.
         if(nx.ne.0) p = real(nx)*2.*pi/alenx
c     Numerical Solution
         chi(kx)=chi(kx)*alenx/sqrt(2.0*pi)/nptx
         re=chi(kx)
         ri=imag(chi(kx))
         IF(re.NE.0) theta=atan(ri/re)
         rm=abs(chi(kx))
         IF(abs(p).LE.(4.)) WRITE(2,22) p,rm,theta
         IF(nx.EQ.(nptx/2)) WRITE(2,22)
c     Analytic Solution
         CALL FT_analy(Psip,p)
         re=Psip
         ri=imag(Psip)
         IF(re.NE.0) theta=atan(ri/re)
         rm=abs(Psip)
         IF(abs(p).LE.(4.)) WRITE(1,22) p,rm,theta
         IF(nx.EQ.(nptx/2)) WRITE(1,22)
      end do
      CALL fourn(chi,nptx,1,1)
 22   FORMAT(6(e13.6,2x))
      return
      end
ccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccc
      subroutine FT_analy(Psip,p)
c
c     Analytic Fourier Transform of the initial Gaussian wave-packet
c
      IMPLICIT NONE
      REAL p,pi,alpha,rmass,xk,pk,omega
      COMPLEX Psip,c0,c1,c2,eye
      common /packet/  rmass,xk,pk
      eye=(0.0,1.0)
      omega=1.
      alpha = rmass*omega
      pi=acos(-1.0)
      c2=alpha/2.
      c1=alpha*xk+eye*(pk-p)
      c0=-alpha/2.*xk**2-eye*pk*xk
      Psip=sqrt(pi/c2)/sqrt(2.0*pi)*(alpha/pi)**0.25
     1     *exp(c1**2/(4.0*c2))*exp(c0)
      return
      end
ccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccc
c     Subroutines from Numerical Recipes
ccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccc
      SUBROUTINE FOURN(DATA,NN,NDIM,ISIGN)
      REAL*8 WR,WI,WPR,WPI,WTEMP,THETA
      DIMENSION NN(NDIM),DATA(*)
      NTOT=1
      DO 11 IDIM=1,NDIM
         NTOT=NTOT*NN(IDIM)
 11   CONTINUE
      NPREV=1
      DO 18 IDIM=1,NDIM
         N=NN(IDIM)
         NREM=NTOT/(N*NPREV)
         IP1=2*NPREV
         IP2=IP1*N
         IP3=IP2*NREM
```

```
        I2REV=1
        DO 14 I2=1,IP2,IP1
           IF(I2.LT.I2REV)THEN
              DO 13 I1=I2,I2+IP1-2,2
                 DO 12 I3=I1,IP3,IP2
                    I3REV=I2REV+I3-I2
                    TEMPR=DATA(I3)
                    TEMPI=DATA(I3+1)
                    DATA(I3)=DATA(I3REV)
                    DATA(I3+1)=DATA(I3REV+1)
                    DATA(I3REV)=TEMPR
                    DATA(I3REV+1)=TEMPI
 12              CONTINUE
 13           CONTINUE
           ENDIF
           IBIT=IP2/2
 1         IF ((IBIT.GE.IP1).AND.(I2REV.GT.IBIT)) THEN
              I2REV=I2REV-IBIT
              IBIT=IBIT/2
              GO TO 1
           ENDIF
           I2REV=I2REV+IBIT
 14      CONTINUE
         IFP1=IP1
 2       IF(IFP1.LT.IP2)THEN
           IFP2=2*IFP1
           THETA=ISIGN*6.28318530717959D0/(IFP2/IP1)
           WPR=-2.D0*DSIN(0.5D0*THETA)**2
           WPI=DSIN(THETA)
           WR=1.D0
           WI=0.D0
           DO 17 I3=1,IFP1,IP1
              DO 16 I1=I3,I3+IP1-2,2
                 DO 15 I2=I1,IP3,IFP2
                    K1=I2
                    K2=K1+IFP1
                    TEMPR=SNGL(WR)*DATA(K2)-SNGL(WI)*DATA(K2+1)
                    TEMPI=SNGL(WR)*DATA(K2+1)+SNGL(WI)*DATA(K2)
                    DATA(K2)=DATA(K1)-TEMPR
                    DATA(K2+1)=DATA(K1+1)-TEMPI
                    DATA(K1)=DATA(K1)+TEMPR
                    DATA(K1+1)=DATA(K1+1)+TEMPI
 15              CONTINUE
 16           CONTINUE
              WTEMP=WR
              WR=WR*WPR-WI*WPI+WR
              WI=WI*WPR+WTEMP*WPI+WI
 17        CONTINUE
           IFP1=IFP2
           GO TO 2
         ENDIF
         NPREV=N*NPREV
 18   CONTINUE
      RETURN
      END
cccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccc
```

Problem 3:

In order to visualize the output of this program, cut the source code attached below, save it in a file named Problem3.f, compile it by typing

```
f77 Problem3.f -o Problem3
```

run it by typing

```
./Problem3
```

The printout on the screen includes the numerically expectation values $\langle\Psi_t|\hat{V}|\Psi_t\rangle$ and $\langle\Psi_t|\hat{x}|\Psi_t\rangle$.

```
      PROGRAM Problem3
      IMPLICIT NONE
      REAL x,VENERGY
      CALL Initialize()
      CALL PE(VENERGY)
      CALL Px(x)
      PRINT *, "<Psi|V|Psi>=",VENERGY
      PRINT *, "<Psi|x|Psi>=",x
      END
ccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccc
      SUBROUTINE Initialize()
c
c     Wave Packet Initialization: Gaussian centered at xk, with momentum pk
c
      IMPLICIT NONE
      INTEGER nptx,npts,kk
      COMPLEX chi,EYE
      REAL omega,xmin,xmax,dx,pi,mass,xk,pk,x,alpha
      PARAMETER(npts=10,nptx=2**npts)
      COMMON / wfunc/ chi(nptx)
      common /xy/  xmin,xmax
      common /packet/mass,xk,pk
      xmin=-20.
      xmax=20.
      EYE=(0.0,1.0)
      pi= acos(-1.0)
      omega=1.
      dx=(xmax-xmin)/real(nptx)
      pk=0.0
      xk=0.0
      mass=1.0
      alpha=mass*omega
      do kk=1,nptx
         x=xmin+kk*dx
         chi(kk)=((alpha/pi)**0.25)
     1        *exp(-alpha/2.*(x-xk)**2+EYE*pk*(x-xk))
      end do
      RETURN
      END
ccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccc
      SUBROUTINE PE(RV)
c
c     Expectation Value of the Potential Enegy
c
      IMPLICIT NONE
      INTEGER nptx,npts,k
      COMPLEX chi
      REAL Vpot,RV,xmin,xmax,dx,x
      PARAMETER(npts=10,nptx=2**npts)
      COMMON / wfunc/ chi(nptx)
      common /xy/  xmin,xmax
      dx=(xmax-xmin)/real(nptx)
      RV=0.0
      do k=1,nptx
         x=xmin+k*dx
         CALL VA(Vpot,x)
         RV=RV+chi(k)*Vpot*conjg(chi(k))*dx
      end do
      RETURN
      END
ccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccc
```

```
      SUBROUTINE Px(RV)
c
c     Expectation Value of the position
c
      IMPLICIT NONE
      INTEGER nptx,npts,k
      COMPLEX chi
      REAL RV,xmin,xmax,dx,x
      PARAMETER(npts=10,nptx=2**npts)
      COMMON / wfunc/ chi(nptx)
      common /xy/  xmin,xmax
      dx=(xmax-xmin)/real(nptx)
      RV=0.0
      do k=1,nptx
         x=xmin+k*dx
         RV=RV+chi(k)*x*conjg(chi(k))*dx
      end do
      RETURN
      END
ccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccc
      SUBROUTINE VA(V,x)
c
c     Potential Energy Surface: Harmonic Oscillator
c
      IMPLICIT NONE
      REAL V,x,mass,xk,pk,rk,omega
      common /packet/ mass,xk,pk
      omega=1.0
      rk=mass*omega**2
      V=0.5*rk*x*x
      RETURN
      END
ccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccc
```

Problem 4:

In order to visualize the output of this program, cut the source code attached below, save it in a file named Problem4.f, compile it by typing

```
f77 Problem4.f -o Problem4
```

run it by typing

```
./Problem4
```

The printout on the screen includes the numerically expectation values $\langle\Psi_t|\hat{p}|\Psi_t\rangle$, $\langle\Psi_t|\hat{T}|\Psi_t\rangle$ and $\langle\Psi_t|\hat{H}|\Psi_t\rangle$. Note that the analytic value of $\langle\Psi_t|\hat{T}|\Psi_t\rangle$ is $\hbar\omega/2 = 0.5$ in agreement with the numerical solution.

```
      PROGRAM Problem4
      CALL Initialize()
      CALL Pp(p)
      PRINT *, "<Psi|p|Psi>=",p
      CALL KE(RKE)
      PRINT *, "<Psi|T|Psi>=",RKE
      CALL PE(RV)
      PRINT *, "<Psi|H|Psi>=",RKE+RV
      END
ccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccc
      SUBROUTINE Initialize()
c
c     Wave Packet Initialization: Gaussian centered at xk, with momentum pk
c
      IMPLICIT NONE
      INTEGER nptx,npts,kk
      COMPLEX chi,EYE
      REAL omega,xmin,xmax,dx,pi,mass,xk,pk,x,alpha
      PARAMETER(npts=10,nptx=2**npts)
      COMMON / wfunc/ chi(nptx)
      common /xy/  xmin,xmax
      common /packet/mass,xk,pk
      xmin=-20.
      xmax=20.
      EYE=(0.0,1.0)
      pi= acos(-1.0)
      omega=1.
      dx=(xmax-xmin)/real(nptx)
      pk=0.0
      xk=0.0
      mass=1.0
      alpha=mass*omega
      do kk=1,nptx
         x=xmin+kk*dx
         chi(kk)=((alpha/pi)**0.25)
     1        *exp(-alpha/2.*(x-xk)**2+EYE*pk*(x-xk))
      end do
      RETURN
      END
ccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccc
      SUBROUTINE PE(RV)
c
c     Expectation Value of the Potential Enegy
c
      IMPLICIT NONE
      INTEGER nptx,npts,k
      COMPLEX chi
      REAL Vpot,RV,xmin,xmax,dx,x
      PARAMETER(npts=10,nptx=2**npts)
      COMMON / wfunc/ chi(nptx)
      common /xy/  xmin,xmax
      dx=(xmax-xmin)/real(nptx)
      RV=0.0
      do k=1,nptx
         x=xmin+k*dx
         CALL VA(Vpot,x)
         RV=RV+chi(k)*Vpot*conjg(chi(k))*dx
      end do
      RETURN
      END
ccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccc
```

```
      SUBROUTINE KE(RKE)
c
c     Expectation value of the kinetic energy
c
      IMPLICIT NONE
      INTEGER kk,nptx,kx,nx,npts
      REAL dp,RKE,p,xmin,xmax,pi,alenx,dx,mass,xk,pk
      COMPLEX eye,chi,Psip,chic
      parameter(npts=10,nptx=2**npts)
      DIMENSION chic(nptx)
      common /xy/  xmin,xmax
      common /packet/mass,xk,pk
      COMMON / wfunc/ chi(nptx)
      RKE=0.0
      pi = acos(-1.0)
      dx=(xmax-xmin)/nptx
      dp=2.*pi/(xmax-xmin)
      do kk=1,nptx
         chic(kk)=chi(kk)
      end do
      CALL fourn(chic,nptx,1,1)
      do kx=1,nptx
         if(kx.le.(nptx/2+1)) then
            nx=kx-1
         else
            nx=kx-1-nptx
         end if
         p=0.
         if(nx.ne.0) p = real(nx)*dp
         chic(kx)=p**2/(2.0*mass)*chic(kx)/nptx
      end do
      CALL fourn(chic,nptx,1,-1)
      do kk=1,nptx
         RKE=RKE+conjg(chi(kk))*chic(kk)*dx
      end do
      return
      end
ccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccc
      SUBROUTINE Pp(pe)
c
c     Expectation value of the momentum
c
      IMPLICIT NONE
      INTEGER kk,nptx,kx,nx,npts
      REAL dp,pe,p,xmin,xmax,pi,alenx,dx,mass,xk,pk
      COMPLEX eye,chi,Psip,chic
      parameter(npts=10,nptx=2**npts)
      DIMENSION chic(nptx)
      common /xy/  xmin,xmax
      common /packet/mass,xk,pk
      COMMON / wfunc/ chi(nptx)
      pe=0.0
      pi = acos(-1.0)
      dx=(xmax-xmin)/nptx
      dp=2.*pi/(xmax-xmin)
      do kk=1,nptx
         chic(kk)=chi(kk)
      end do
      CALL fourn(chic,nptx,1,1)
      do kx=1,nptx
         if(kx.le.(nptx/2+1)) then
```

```
            nx=kx-1
         else
            nx=kx-1-nptx
         end if
         p=0.
         if(nx.ne.0) p = real(nx)*dp
         chic(kx)=p*chic(kx)/nptx
      end do
      CALL fourn(chic,nptx,1,-1)
      do kk=1,nptx
         pe=pe+conjg(chi(kk))*chic(kk)*dx
      end do
      return
      end
cccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccc
      SUBROUTINE VA(V,x)
c
c     Potential Energy Surface: Harmonic Oscillator
c
      implicit none
      REAL V,x,mass,xk,pk,rk,omega
      common /packet/ mass,xk,pk
      omega=1.0
      rk=mass*omega**2
      V=0.5*rk*x*x
      RETURN
      END
cccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccc
c     Subroutines from Numerical Recipes
cccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccc
      SUBROUTINE FOURN(DATA,NN,NDIM,ISIGN)
      REAL*8 WR,WI,WPR,WPI,WTEMP,THETA
      DIMENSION NN(NDIM),DATA(*)
      NTOT=1
      DO 11 IDIM=1,NDIM
         NTOT=NTOT*NN(IDIM)
 11   CONTINUE
      NPREV=1
      DO 18 IDIM=1,NDIM
         N=NN(IDIM)
         NREM=NTOT/(N*NPREV)
         IP1=2*NPREV
         IP2=IP1*N
         IP3=IP2*NREM
         I2REV=1
         DO 14 I2=1,IP2,IP1
            IF(I2.LT.I2REV)THEN
               DO 13 I1=I2,I2+IP1-2,2
                  DO 12 I3=I1,IP3,IP2
                     I3REV=I2REV+I3-I2
                     TEMPR=DATA(I3)
                     TEMPI=DATA(I3+1)
                     DATA(I3)=DATA(I3REV)
                     DATA(I3+1)=DATA(I3REV+1)
                     DATA(I3REV)=TEMPR
                     DATA(I3REV+1)=TEMPI
 12               CONTINUE
 13            CONTINUE
            ENDIF
            IBIT=IP2/2
 1          IF ((IBIT.GE.IP1).AND.(I2REV.GT.IBIT)) THEN
```

```
               I2REV=I2REV-IBIT
               IBIT=IBIT/2
               GO TO 1
            ENDIF
            I2REV=I2REV+IBIT
 14      CONTINUE
         IFP1=IP1
 2       IF(IFP1.LT.IP2)THEN
            IFP2=2*IFP1
            THETA=ISIGN*6.28318530717959D0/(IFP2/IP1)
            WPR=-2.D0*DSIN(0.5D0*THETA)**2
            WPI=DSIN(THETA)
            WR=1.D0
            WI=0.D0
            DO 17 I3=1,IFP1,IP1
               DO 16 I1=I3,I3+IP1-2,2
                  DO 15 I2=I1,IP3,IFP2
                     K1=I2
                     K2=K1+IFP1
                     TEMPR=SNGL(WR)*DATA(K2)-SNGL(WI)*DATA(K2+1)
                     TEMPI=SNGL(WR)*DATA(K2+1)+SNGL(WI)*DATA(K2)
                     DATA(K2)=DATA(K1)-TEMPR
                     DATA(K2+1)=DATA(K1+1)-TEMPI
                     DATA(K1)=DATA(K1)+TEMPR
                     DATA(K1+1)=DATA(K1+1)+TEMPI
 15               CONTINUE
 16            CONTINUE
               WTEMP=WR
               WR=WR*WPR-WI*WPI+WR
               WI=WI*WPR+WTEMP*WPI+WI
 17         CONTINUE
            IFP1=IFP2
            GO TO 2
         ENDIF
         NPREV=N*NPREV
 18   CONTINUE
      RETURN
      END
cccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccc
```

Problem 5:

Expanding the left-hand-side (l.h.s.) of Eq. (18) from the lecture notes gives:

$$e^{-i\hat{H}\tau} = 1 - i\hat{H}\tau - \frac{1}{2}\hat{H}^2\tau^2 + O(\tau^3), \tag{1}$$

where $\hat{H} = \hat{p}^2/(2m) + \hat{V}$. Therefore,

$$e^{-i\hat{H}\tau} = 1 - i\hat{H}\tau - \frac{1}{2}\frac{\hat{p}^4}{4m^2}\tau^2 - \frac{1}{2}\hat{V}^2\tau^2 - \frac{1}{2}\frac{\hat{p}^2}{2m}\hat{V}\tau^2 - \frac{1}{2}\hat{V}\frac{\hat{p}^2}{2m}\tau^2 + O(\tau^3), \tag{2}$$

In order to show that the Trotter expansion, introduced by Eq. (18), is accurate to second order in τ, we must expand the right-hand-side (r.h.s.) of Eq. (18) and show that suchh an expansion equals the r.h.s. of Eq. (2).

Expanding the right-hand-side (r.h.s.) of Eq. (18) gives,

$$\begin{aligned} e^{-iV(\hat{x})\tau/2}e^{-i\hat{p}^2\tau/(2m)}e^{-iV(\hat{x})\tau/2} = &\left(1 - i\hat{V}\tau/2 - \frac{1}{2}\hat{V}^2\tau^2/4 + O(\tau^3)\right)\left(1 - i\frac{\hat{p}^2}{2m}\tau - \frac{1}{2}\frac{\hat{p}^4}{4m^2}\tau^2 + O(\tau^3)\right) \\ &\times \left(1 - i\hat{V}\tau/2 - \frac{1}{2}\hat{V}^2\tau^2/4 + O(\tau^3)\right), \end{aligned} \tag{3}$$

$$\begin{aligned} e^{-iV(\hat{x})\tau/2}e^{-i\hat{p}^2\tau/(2m)}e^{-iV(\hat{x})\tau/2} = &\left(1 - i\hat{V}\tau/2 - \frac{1}{2}\hat{V}^2\tau^2/4 - i\frac{\hat{p}^2}{2m}\tau - \hat{V}\frac{\hat{p}^2}{2m}\tau^2/2 - \frac{1}{2}\frac{\hat{p}^4}{4m^2}\tau^2 + O(\tau^3)\right) \\ &\times \left(1 - i\hat{V}\tau/2 - \frac{1}{2}\hat{V}^2\tau^2/4 + O(\tau^3)\right), \end{aligned} \tag{4}$$

$$\begin{aligned} e^{-iV(\hat{x})\tau/2}e^{-i\hat{p}^2\tau/(2m)}e^{-iV(\hat{x})\tau/2} = &1 - i\hat{V}\tau/2 - \frac{1}{2}\hat{V}^2\tau^2/4 - i\frac{\hat{p}^2}{2m}\tau - \hat{V}\frac{\hat{p}^2}{2m}\tau^2/2 - \frac{1}{2}\frac{\hat{p}^4}{4m^2}\tau^2 \\ &- i\hat{V}\tau/2 - \hat{V}^2\tau^2/4 - \frac{\hat{p}^2}{2m}\hat{V}\tau^2/2 - \frac{1}{2}\hat{V}^2\tau^2/4 + O(\tau^3), \end{aligned} \tag{5}$$

$$\begin{aligned} e^{-iV(\hat{x})\tau/2}e^{-i\hat{p}^2\tau/(2m)}e^{-iV(\hat{x})\tau/2} = &1 - i\hat{V}\tau - i\frac{\hat{p}^2}{2m}\tau - \frac{1}{2}\hat{V}^2\tau^2 - \hat{V}\frac{\hat{p}^2}{2m}\tau^2/2 - \frac{1}{2}\frac{\hat{p}^4}{4m^2}\tau^2 \\ &- \frac{\hat{p}^2}{2m}\hat{V}\tau^2/2 + O(\tau^3). \end{aligned} \tag{6}$$

Note that the r.h.s. of Eq. (6) is identical to the r.h.s. of E. (2), completing the proof that the Trotter expansion, introduced by Eq. (18), is accurate to second order in τ.

Problem 6:

In order to visualize the output of this program, cut the source code attached below, save it in a file named Problem6.f, compile it by typing

```
f77 Problem6.f -o Problem6
```

run it by typing

```
./Problem6
```

and visualize the output as follows: type

```
gnuplot
```

then type

```
set dat sty line
```

then type

```
set yrange[0:6]
```

and the type

```
plot ``arch.0002''
```

That will show the numerical propagation after one step with $\tau = 0.1$. In order to visualize the analytic result on top of the numerical propagation, type

```
replot ``arch.0002'' u 1:3
```

To exit, type

```
quit
```

```
      PROGRAM Problem6
c
c     1-D wave packet propagation
c
      IMPLICIT NONE
      INTEGER NN,npts,nptx,ndump
      INTEGER istep,nstep
      REAL dt,xc,pc
      COMPLEX vprop,tprop,x_mean,p_mean
      PARAMETER(npts=9,nptx=2**npts,NN=1)
      DIMENSION vprop(nptx,NN,NN),tprop(nptx)
      DIMENSION x_mean(NN),p_mean(NN)
      COMMON /class/ xc,pc
c
      CALL ReadParam(nstep,ndump,dt)
      call Initialize()
      CALL SetKinProp(dt,tprop)
      CALL SetPotProp(dt,vprop)
      DO istep=1,nstep+1
         IF(mod(istep-1,10).EQ.0)
     1        PRINT *, "Step=", istep-1,", Final step=", nstep
         IF(istep.GE.1) CALL PROPAGATE(vprop,tprop)
         IF(mod((istep-1),ndump).EQ.0) THEN
            CALL SAVEWF(istep,ndump,dt)
         END IF
      END DO
 22   FORMAT(6(e13.6,2x))
      END
ccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccc
      subroutine ReadParam(nstep,ndump,dt)
c
c     Parameters defining the grid (xmin, xmax), integration time step (dt),
c     mass (rmass), initial position (xk), initial momentum (pk),
c     number of propagation steps (nstep), and how often to save a pic (ndump)
c
      IMPLICIT NONE
      INTEGER ntype,nstep,nrpt,ireport,ndump,nlit
      REAL xmin,xmax,pk,rmass,xk,dt
      common /packet/  rmass,xk,pk
      common /xy/  xmin,xmax
c
      xmin=-10.0
      xmax= 10.0
      dt=0.1
      rmass=1.0
      xk=-2.5
      pk=1.0
      nstep=1
      ndump=1
c
      return
      end
ccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccc
      SUBROUTINE Initialize()

      IMPLICIT NONE
      INTEGER NN,nptx,npts,kk
      COMPLEX chi0,chi,EYE,CRV
      REAL xc,pc,omega,xk2,xmin,xmax,dx,pi,rmass,xk,pk,x,alpha,alpha2
      PARAMETER(npts=9,nptx=2**npts,NN=1)
      DIMENSION CRV(NN,NN)
```

```
      common /xy/  xmin,xmax
      common /packet/  rmass,xk,pk
      COMMON / wfunc/ chi(nptx,NN)
      COMMON / iwfunc/ chi0(nptx,NN)
      COMMON /class/ xc,pc

      EYE=(0.0,1.0)
      pi= acos(-1.0)
      omega=1.
      dx=(xmax-xmin)/real(nptx)
      xc=kk
      pc=pk
c
c     Wave Packet Initialization: Gaussian centered at xk, with momentum pk
c
      alpha=rmass*omega
      do kk=1,nptx
         x=xmin+kk*dx
         chi(kk,1)=((alpha/pi)**0.25)
     1        *exp(-alpha/2.*(x-xk)**2+EYE*pk*(x-xk))
         chi0(kk,1)=chi(kk,1)
      end do
c
c     Hamiltonian Matrix CRV
c
      do kk=1,nptx
         x=xmin+kk*dx
         CALL HAMIL(CRV,x)
         WRITE(11,22) x,real(CRV(1,1))
      END DO
 22   FORMAT(6(e13.6,2x))
      RETURN
      END
ccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccc
      SUBROUTINE HAMIL(CRV,x)
c
c     Hamiltonian Matrix
c
      IMPLICIT NONE
      INTEGER NN
      REAL x,VPOT1
      COMPLEX CRV
      PARAMETER(NN=1)
      DIMENSION CRV(NN,NN)
c
      CALL VA(VPOT1,x)
      CRV(1,1)=VPOT1
c
      RETURN
      END
ccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccc
      SUBROUTINE VA(V,x)
c
c     Potential Energy Surface: Harmonic Oscillator
c
      implicit none
      REAL V,x,rmass,xk,pk,rk,omega
      common /packet/ rmass,xk,pk
      omega=1.0
      rk=rmass*omega**2
      V=0.5*rk*x*x
```

```
      RETURN
      END
ccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccc
      subroutine SetKinProp(dt,tprop)
c
c     Kinetic Energy part of the Trotter Expansion: exp(-i p^2 dt/(2 m))
c
      IMPLICIT NONE
      INTEGER nptx,kx,nx,npts
      REAL xsc,xmin,xmax,propfacx,rmass,xk,pi,alenx,dt,pk
      COMPLEX tprop,eye
      parameter(npts=9,nptx=2**npts)
      DIMENSION tprop(nptx)
      common /xy/  xmin,xmax
      common /packet/  rmass,xk,pk
c
      eye=(0.,1.)
      pi = acos(-1.0)
      alenx=xmax-xmin
      propfacx=-dt/2./rmass*(2.*pi)**2
      do kx=1,nptx
         if(kx.le.(nptx/2+1)) then
            nx=kx-1
         else
            nx=kx-1-nptx
         end if
         xsc=0.
         if(nx.ne.0) xsc=real(nx)/alenx
         tprop(kx)=exp(eye*(propfacx*xsc**2))
       end do
c
      return
      end
ccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccc
      subroutine SetPotProp(dt,vprop)
c
c     Potential Energy part of the Trotter Expansion: exp(-i V dt/2)
c
      IMPLICIT NONE
      INTEGER NN,ii,nptx,npts
      REAL xmin,xmax,dx,dt,x,VPOT
      COMPLEX vprop,eye
      parameter(npts=9,nptx=2**npts,NN=1)
      DIMENSION vprop(nptx,NN,NN)
      common /xy/  xmin,xmax
      eye=(0.,1.)
      dx=(xmax-xmin)/real(nptx)
c
      do ii=1,nptx
         x=xmin+ii*dx
         CALL VA(VPOT,x)
         vprop(ii,1,1)=exp(-eye*0.5*dt*VPOT)/sqrt(nptx*1.0)
      END DO
      RETURN
      END
ccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccc
      SUBROUTINE energies(energy)
      IMPLICIT NONE
      INTEGER j,NN
      COMPLEX energy,RV,RKE
      PARAMETER (NN=1)
```

```
      DIMENSION RV(NN),RKE(NN),energy(NN)
      CALL PE(RV)
      CALL KE(RKE)
      DO j=1,NN
         energy(j)=RV(j)+RKE(j)
      END DO
      RETURN
      END
ccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccc
      FUNCTION Psia(x,istep,dt)
c
c     Analytic wave-packet <x|Psia(istep)> obtained by applying the
c     harmonic propagator to the initial state,
c     <x'|Psi(0)> = (alpha/pi)**.25*exp(-alpha/2*(x'-xk)**2+eye*pk*(x'-xk)),
c     where the propagator is
c     <x|exp(-beta H)|x'> = A exp(-rgamma*(x**2+x'**2)+rgammap*x*x'), with
c     A = sqrt(m*omega/(pi*(exp(beta*omega)-exp(-beta*omega)))), beta = i*t,
c     rgamma = 0.5*m*omega*cosh(beta*omega)/sinh(beta*omega) and
c     rgammap = m*omega/sinh(beta*omega).
c
      IMPLICIT NONE
      INTEGER istep
      REAL pk,rmass,xk,dt,x,t,omega,pi,alpha
      COMPLEX eye,Psia,beta,A,rgamma,rgammap,c0,c1,c2
      common /packet/  rmass,xk,pk
      eye=(0.0,1.0)
      omega=1.0
      alpha = omega*rmass
      pi=acos(-1.0)
      beta = eye*dt*istep
      IF(abs(beta).EQ.0) beta = eye*1.0E-7
      A = sqrt(rmass*omega/(pi*(exp(beta*omega)-exp(-beta*omega))))
      rgamma=0.5*rmass*omega*(exp(beta*omega)+exp(-beta*omega))
     1     /(exp(beta*omega)-exp(-beta*omega))
      rgammap=2.*rmass*omega/(exp(beta*omega)-exp(-beta*omega))
      c0=-eye*pk*xk-alpha/2.*xk**2
      c1=rgammap*x+alpha*xk+eye*pk
      c2=rgamma+alpha/2.
c
      Psia = A*(alpha/pi)**.25*sqrt(pi/c2)*
     1     exp(-rgamma*x**2)*exp(c0+c1**2/(4.0*c2))
c
      return
      end
ccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccc
      SUBROUTINE SAVEWF(je2,ndump,dt)
c
c     Dump Time Evolved Wave packet
c
      IMPLICIT NONE
      INTEGER je2,nptx,npts,kk,NN,ncount,ndump,jj
      COMPLEX chi,CRV,energy,psi,Psia
      character*9 B
      REAL V,x1,c1,c2,c1a,x,xmin,xmax,dx,EVALUES,dt
      PARAMETER(npts=9,nptx=2**npts,NN=1)
      DIMENSION CRV(NN,NN),energy(NN),EVALUES(NN)
      DIMENSION psi(NN,NN)
      common /xy/  xmin,xmax
      COMMON / wfunc/ chi(nptx,NN)
c
      CALL energies(energy)
```

```
      jj=je2/ndump
      write(B, '(A,i4.4)') 'arch.', jj
      OPEN(1,FILE=B)
      dx=(xmax-xmin)/real(nptx)
      ncount=(je2-1)/ndump
c
c     Save Wave-packet components
c
      do kk=1,nptx
         x=xmin+kk*dx
         c1=chi(kk,1)*conjg(chi(kk,1))
         c1a=Psia(x,je2,dt)*conjg(Psia(x,je2,dt))
         write(1,33) x,sqrt(c1)+real(energy(1))
     1        ,sqrt(c1a)+real(energy(1))
      end do
      write(1,33)
      do kk=1,nptx
         x=xmin+kk*dx
         write(1,33) x
     1        ,real(chi(kk,1))+real(energy(1))
     1        ,real(Psia(x,je2,dt))+real(energy(1))
      end do
      write(1,33)
c
c     Save Adiabatic states
c
      do kk=1,nptx
         x=xmin+kk*dx
         CALL HAMIL(CRV,x)
         write(1,33) x,CRV(1,1)
      end do
      CLOSE(1)
 33   format(6(e13.6,2x))
      RETURN
      END
ccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccc
      SUBROUTINE PE(RV)
c
c     Expectation Value of the Potential Enegy
c
      IMPLICIT NONE
      INTEGER nptx,npts,kk,NN,j
      COMPLEX chi,EYE,RV
      REAL Vpot,omega,xmin,xmax,dx,pi,rmass,xk,pk,x,alpha
      PARAMETER(npts=9,nptx=2**npts,NN=1)
      DIMENSION RV(NN)
      COMMON / wfunc/ chi(nptx,NN)
      common /xy/  xmin,xmax
      common /packet/rmass,xk,pk

      dx=(xmax-xmin)/real(nptx)
      DO j=1,NN
         RV(j)=0.0
         do kk=1,nptx
            x=xmin+kk*dx
            IF(j.EQ.1) CALL VA(Vpot,x)
            RV(j)=RV(j)+chi(kk,j)*Vpot*conjg(chi(kk,j))*dx
         end do
      END DO
      RETURN
      END
```

```
ccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccc
      subroutine KE(RKE)
c
c     Expectation value of the kinetic energy
c
      IMPLICIT NONE
      INTEGER NN,kk,nptx,kx,nx,npts,j
      REAL dp,theta,wm,p,xmin,xmax,rmass,xk,pi,alenx,pk,rm,re,ri,dx
      COMPLEX eye,chi,Psip,chic,RKE
      parameter(npts=9,nptx=2**npts,NN=1)
      DIMENSION chic(nptx),RKE(NN)
      common /xy/  xmin,xmax
      common /packet/  rmass,xk,pk
      COMMON / wfunc/ chi(nptx,NN)
c
      pi = acos(-1.0)
      dx=(xmax-xmin)/nptx
      dp=2.*pi/(xmax-xmin)
c
      DO j=1,NN
         RKE(j)=0.0
         do kk=1,nptx
            chic(kk)=chi(kk,j)
         end do
         CALL fourn(chic,nptx,1,-1)
         do kx=1,nptx
            if(kx.le.(nptx/2+1)) then
               nx=kx-1
            else
               nx=kx-1-nptx
            end if
            p=0.
            if(nx.ne.0) p = real(nx)*dp
            chic(kx)=p**2/(2.0*rmass)*chic(kx)/nptx
         end do
         CALL fourn(chic,nptx,1,1)
         do kk=1,nptx
            RKE(j)=RKE(j)+conjg(chi(kk,j))*chic(kk)*dx
         end do
      END DO
      return
      end
ccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccc
      SUBROUTINE PROPAGATE(vprop,tprop)
c
c     Split Operator Fourier Transform Propagation Method
c     J. Comput. Phys. 47, 412 (1982); J. Chem. Phys. 78, 301 (1983)
c
      IMPLICIT NONE
      INTEGER i,j,NN,ii,nptx,npts
      COMPLEX chi,vprop,chin1,chin2,tprop
      PARAMETER(npts=9,nptx=2**npts,NN=1)
      DIMENSION chin1(nptx),chin2(nptx)
      DIMENSION tprop(nptx),vprop(nptx,NN,NN)
      COMMON / wfunc/ chi(nptx,NN)
c
c     Apply potential energy part of the Trotter Expansion
c
      DO i=1,nptx
         chin1(i)=0.0
         DO j=1,NN
```

```
            chin1(i)=chin1(i)+vprop(i,1,j)*chi(i,j)
         END DO
      END DO
c
c     Fourier Transform wave-packet to the momentum representation
c
      CALL fourn(chin1,nptx,1,-1)
c
c     Apply kinetic energy part of the Trotter Expansion
c
      DO i=1,nptx
         chin1(i)=tprop(i)*chin1(i)
      END DO
c
c     Inverse Fourier Transform wave-packet to the coordinate representation
c
      CALL fourn(chin1,nptx,1,1)
c
c     Apply potential energy part of the Trotter Expansion
c
      DO i=1,nptx
         DO j=1,NN
            chi(i,j)=vprop(i,j,1)*chin1(i)
         END DO
      END DO
      END
ccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccc
c     Subroutine for FFT from Numerical Recipes
ccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccc
      SUBROUTINE FOURN(DATA,NN,NDIM,ISIGN)
      REAL*8 WR,WI,WPR,WPI,WTEMP,THETA
      DIMENSION NN(NDIM),DATA(*)
      NTOT=1
      DO 11 IDIM=1,NDIM
         NTOT=NTOT*NN(IDIM)
 11   CONTINUE
      NPREV=1
      DO 18 IDIM=1,NDIM
         N=NN(IDIM)
         NREM=NTOT/(N*NPREV)
         IP1=2*NPREV
         IP2=IP1*N
         IP3=IP2*NREM
         I2REV=1
         DO 14 I2=1,IP2,IP1
            IF(I2.LT.I2REV)THEN
               DO 13 I1=I2,I2+IP1-2,2
                  DO 12 I3=I1,IP3,IP2
                     I3REV=I2REV+I3-I2
                     TEMPR=DATA(I3)
                     TEMPI=DATA(I3+1)
                     DATA(I3)=DATA(I3REV)
                     DATA(I3+1)=DATA(I3REV+1)
                     DATA(I3REV)=TEMPR
                     DATA(I3REV+1)=TEMPI
 12               CONTINUE
 13            CONTINUE
            ENDIF
            IBIT=IP2/2
 1          IF ((IBIT.GE.IP1).AND.(I2REV.GT.IBIT)) THEN
               I2REV=I2REV-IBIT
```

```
               IBIT=IBIT/2
               GO TO 1
            ENDIF
            I2REV=I2REV+IBIT
 14      CONTINUE
         IFP1=IP1
 2       IF(IFP1.LT.IP2)THEN
            IFP2=2*IFP1
            THETA=ISIGN*6.28318530717959D0/(IFP2/IP1)
            WPR=-2.D0*DSIN(0.5D0*THETA)**2
            WPI=DSIN(THETA)
            WR=1.D0
            WI=0.D0
            DO 17 I3=1,IFP1,IP1
               DO 16 I1=I3,I3+IP1-2,2
                  DO 15 I2=I1,IP3,IFP2
                     K1=I2
                     K2=K1+IFP1
                     TEMPR=SNGL(WR)*DATA(K2)-SNGL(WI)*DATA(K2+1)
                     TEMPI=SNGL(WR)*DATA(K2+1)+SNGL(WI)*DATA(K2)
                     DATA(K2)=DATA(K1)-TEMPR
                     DATA(K2+1)=DATA(K1+1)-TEMPI
                     DATA(K1)=DATA(K1)+TEMPR
                     DATA(K1+1)=DATA(K1+1)+TEMPI
 15               CONTINUE
 16            CONTINUE
               WTEMP=WR
               WR=WR*WPR-WI*WPI+WR
               WI=WI*WPR+WTEMP*WPI+WI
 17         CONTINUE
            IFP1=IFP2
            GO TO 2
         ENDIF
         NPREV=N*NPREV
 18   CONTINUE
      RETURN
      END
cccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccc
```

Problem 7:

In order to visualize the output of this program, cut the source code attached bellow, save it in a file named Problem7.f, compile it by typing

```
f77 Problem7.f -o Problem7
```

run it by typing

```
./Problem7
```

Visualize the output of time dependent expectation values as compared to classical trajectories as follows: type

```
gnuplot
```

then type

```
set dat sty line
```

then type

```
plot ''traj.0000''
```

That will show the numerical computation of the expectation value $< \Psi_t|\hat{x}|\Psi_t >$ as a function of time. In order to visualize the classical result on top of the quantum mechanical expectation value, type

```
replot ''traj.0000'' u 1:4
```

In order to visualize the output of $< \Psi_t|\hat{p}|\Psi_t >$ as a function of time, type

```
plot ''traj.0000'' u 1:3
```

and to visualize the classical result on top of the quantum mechanical expectation value, type

```
replot ''traj.0000'' u 1:5
```

The plot of $< \Psi_t|\hat{p}|\Psi_t >$ vs. $< \Psi_t|\hat{x}|\Psi_t >$ can be obtained by typing

```
plot ''traj.0000'' u 3:2
```

, and the corresponding classical results $p(t)$ vs. $x(t)$

```
plot ''traj.0000'' u 5:4
```

To exit, type

```
quit
```

The snapshots of the time-dependent wave-packet can be visualized as a movie by typing

```
gnuplot<pp_7
```

where the file named

```
pp_7
```

has the following lines:

```
set yrange[0:6]
set xrange[-10:10]
set dat sty l
plot "arch.0001" u 1:2 lw 3
pause .1
plot "arch.0002" u 1:2 lw 3
pause .1
plot "arch.0003" u 1:2 lw 3
pause .1
plot "arch.0004" u 1:2 lw 3
pause .1
plot "arch.0005" u 1:2 lw 3
pause .1
plot "arch.0006" u 1:2 lw 3
pause .1
plot "arch.0007" u 1:2 lw 3
pause .1
plot "arch.0008" u 1:2 lw 3
pause .1
plot "arch.0009" u 1:2 lw 3
pause .1
plot "arch.0010" u 1:2 lw 3
pause .1
plot "arch.0011" u 1:2 lw 3
pause .1
plot "arch.0012" u 1:2 lw 3
pause .1
plot "arch.0013" u 1:2 lw 3
pause .1
plot "arch.0014" u 1:2 lw 3
pause .1
plot "arch.0015" u 1:2 lw 3
pause .1
plot "arch.0016" u 1:2 lw 3
pause .1
plot "arch.0017" u 1:2 lw 3
pause .1
plot "arch.0018" u 1:2 lw 3
pause .1
plot "arch.0019" u 1:2 lw 3
pause .1
plot "arch.0020" u 1:2 lw 3
pause .1
plot "arch.0021" u 1:2 lw 3
pause .1
plot "arch.0022" u 1:2 lw 3
pause .1
plot "arch.0023" u 1:2 lw 3
pause .1
plot "arch.0024" u 1:2 lw 3
pause .1
plot "arch.0025" u 1:2 lw 3
pause .1
plot "arch.0026" u 1:2 lw 3
pause .1
plot "arch.0027" u 1:2 lw 3
pause .1
plot "arch.0028" u 1:2 lw 3
pause .1
plot "arch.0029" u 1:2 lw 3
pause .1
```

```
plot "arch.0030" u 1:2 lw 3
pause .1
plot "arch.0031" u 1:2 lw 3
pause .1
plot "arch.0032" u 1:2 lw 3
pause .1
plot "arch.0033" u 1:2 lw 3
pause .1
plot "arch.0034" u 1:2 lw 3
pause .1
plot "arch.0035" u 1:2 lw 3
pause .1
plot "arch.0036" u 1:2 lw 3
pause .1
plot "arch.0037" u 1:2 lw 3
pause .1
plot "arch.0038" u 1:2 lw 3
pause .1
plot "arch.0039" u 1:2 lw 3
pause .1
plot  "arch.0040" u 1:2 lw 3
pause .1
plot "arch.0041" u 1:2 lw 3
pause .1
plot "arch.0042" u 1:2 lw 3
pause .1
plot "arch.0043" u 1:2 lw 3
pause .1
plot "arch.0044" u 1:2 lw 3
pause .1
plot "arch.0045" u 1:2 lw 3
pause .1
plot "arch.0046" u 1:2 lw 3
pause .1
plot "arch.0047" u 1:2 lw 3
pause .1
plot "arch.0048" u 1:2 lw 3
pause .1
plot "arch.0049" u 1:2 lw 3
pause .1
plot "arch.0050" u 1:2 lw 3
pause .1
plot "arch.0051" u 1:2 lw 3
pause .1
plot "arch.0052" u 1:2 lw 3
pause .1
plot "arch.0053" u 1:2 lw 3
pause .1
plot "arch.0054" u 1:2 lw 3
pause .1
plot "arch.0055" u 1:2 lw 3
pause .1
plot "arch.0056" u 1:2 lw 3
pause .1
plot "arch.0057" u 1:2 lw 3
pause .1
plot "arch.0058" u 1:2 lw 3
pause .1
plot "arch.0059" u 1:2 lw 3
pause .1
plot "arch.0060" u 1:2 lw 3
```

```
pause .1
plot "arch.0061" u 1:2 lw 3
pause .1
plot "arch.0062" u 1:2 lw 3
pause .1
plot "arch.0063" u 1:2 lw 3
pause .1
plot "arch.0064" u 1:2 lw 3
pause .1
plot "arch.0065" u 1:2 lw 3
pause .1
plot "arch.0066" u 1:2 lw 3
pause .1
plot "arch.0067" u 1:2 lw 3
pause .1
plot "arch.0068" u 1:2 lw 3
pause .1
plot "arch.0069" u 1:2 lw 3
pause .1
plot "arch.0070" u 1:2 lw 3
pause .1
plot "arch.0071" u 1:2 lw 3
pause .1
plot "arch.0072" u 1:2 lw 3
pause .1
plot "arch.0073" u 1:2 lw 3
pause .1
plot "arch.0074" u 1:2 lw 3
pause .1
plot "arch.0075" u 1:2 lw 3
pause .1
plot "arch.0076" u 1:2 lw 3
pause .1
plot "arch.0077" u 1:2 lw 3
pause .1
plot "arch.0078" u 1:2 lw 3
pause .1
plot "arch.0079" u 1:2 lw 3
pause .1
plot  "arch.0080" u 1:2 lw 3
pause .1
plot "arch.0081" u 1:2 lw 3
pause .1
plot "arch.0082" u 1:2 lw 3
pause .1
plot "arch.0083" u 1:2 lw 3
pause .1
plot "arch.0084" u 1:2 lw 3
pause .1
plot "arch.0085" u 1:2 lw 3
pause .1
plot "arch.0086" u 1:2 lw 3
pause .1
plot "arch.0087" u 1:2 lw 3
pause .1
plot "arch.0088" u 1:2 lw 3
pause .1
plot "arch.0089" u 1:2 lw 3
pause .1
plot  "arch.0090" u 1:2 lw 3
pause .1
```

```
plot "arch.0091" u 1:2 lw 3
pause .1
plot "arch.0092" u 1:2 lw 3
pause .1
plot "arch.0093" u 1:2 lw 3
pause .1
plot "arch.0094" u 1:2 lw 3
pause .1
plot "arch.0095" u 1:2 lw 3
pause .1
plot "arch.0096" u 1:2 lw 3
pause .1
plot "arch.0097" u 1:2 lw 3
pause .1
plot "arch.0098" u 1:2 lw 3
pause .1
plot "arch.0099" u 1:2 lw 3
pause .1
```

```
      PROGRAM Problem7
c
c     1-D wave packet propagation and Velocity-Verlet propagation
c     on a Harmonic potential energy surface
c
      IMPLICIT NONE
      INTEGER NN,npts,nptx,ndump
      INTEGER istep,nstep,jj
      REAL dt,xc,pc
      COMPLEX vprop,tprop,x_mean,p_mean
      character*9 Bfile
      PARAMETER(npts=9,nptx=2**npts,NN=1)
      DIMENSION vprop(nptx,NN,NN),tprop(nptx)
      DIMENSION x_mean(NN),p_mean(NN)
      COMMON /class/ xc,pc
c
      jj=0
      write(Bfile, '(A,i4.4)') 'traj.', jj
      OPEN(10,FILE=Bfile)
      CALL ReadParam(nstep,ndump,dt)
      call Initialize()
      CALL SetKinProp(dt,tprop)
      CALL SetPotProp(dt,vprop)
      DO istep=1,nstep+1
         IF(mod(istep-1,10).EQ.0)
     1        PRINT *, "Step=", istep-1,", Final step=", nstep
         IF(istep.GE.1) CALL PROPAGATE(vprop,tprop)
         IF(mod((istep-1),ndump).EQ.0) THEN
            CALL SAVEWF(istep,ndump,dt)
            CALL XM(x_mean)
            CALL PM(p_mean)
            CALL VV(dt)
            WRITE(10,22) (istep-1.)*dt
     1           ,real(x_mean(1)),real(p_mean(1)),xc,pc
         END IF
      END DO
      CLOSE(10)
 22   FORMAT(6(e13.6,2x))
      END
ccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccc
      subroutine ReadParam(nstep,ndump,dt)
c
c     Parameters defining the grid (xmin, xmax), integration time step (dt),
c     rmass (rmass), initial position (xk), initial momentum (pk),
c     number of propagation steps (nstep), and how often to save a pic (ndump)
c
      IMPLICIT NONE
      INTEGER ntype,nstep,nrpt,ireport,ndump,nlit
      REAL xmin,xmax,pk,rmass,xk,dt
      common /packet/  rmass,xk,pk
      common /xy/  xmin,xmax
c
      xmin=-10.0
      xmax= 10.0
      dt=0.1
      rmass=1.0
      xk=-2.5
      pk=0.0
      nstep=100
      ndump=1
c
```

```
      return
      end
ccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccc
      SUBROUTINE VV(dt)
c
c     Velocity Verlet Algorithm J. Chem. Phys. 76, 637 (1982)
c
      IMPLICIT NONE
      REAL v,dx,dt,xc,pc,rmass,xk,pk,acc,xt,VPOT1,VPOT2,F
      COMMON /class/ xc,pc
      common /packet/  rmass,xk,pk
c
c     Compute Force
c
      dx=0.01
      xt=xc+dx
      CALL VA(VPOT1,xt)
      xt=xc-dx
      CALL VA(VPOT2,xt)
      F=-(VPOT1-VPOT2)/(2.0*dx)
      v=pc/rmass
c
c     Advance momenta half a step
c
      pc=pc+0.5*F*dt
c
c     Advance coordinates a step
c
      xc=xc+v*dt+0.5*dt**2*F/rmass
c
c     Compute Force
c
      dx=0.01
      xt=xc+dx
      CALL VA(VPOT1,xt)
      xt=xc-dx
      CALL VA(VPOT2,xt)
      F=-(VPOT1-VPOT2)/(2.0*dx)
c
c     Advance momenta half a step
c
      pc=pc+0.5*F*dt
c
      return
      end
ccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccc
      SUBROUTINE Initialize()

      IMPLICIT NONE
      INTEGER NN,nptx,npts,kk
      COMPLEX chi0,chi,EYE,CRV
      REAL xc,pc,omega,xk2,xmin,xmax,dx,pi,rmass,xk,pk,x,alpha,alpha2
      PARAMETER(npts=9,nptx=2**npts,NN=1)
      DIMENSION CRV(NN,NN)
      common /xy/  xmin,xmax
      common /packet/  rmass,xk,pk
      COMMON / wfunc/ chi(nptx,NN)
      COMMON / iwfunc/ chi0(nptx,NN)
      COMMON /class/ xc,pc

      EYE=(0.0,1.0)
```

```
      pi= acos(-1.0)
      omega=1.
      dx=(xmax-xmin)/real(nptx)
      xc=xk
      pc=pk
c
c     Wave Packet Initialization: Gaussian centered at xk, with momentum pk
c
      alpha=rmass*omega
      do kk=1,nptx
         x=xmin+kk*dx
         chi(kk,1)=((alpha/pi)**0.25)
     1        *exp(-alpha/2.*(x-xk)**2+EYE*pk*(x-xk))
         chi0(kk,1)=chi(kk,1)
      end do
      RETURN
      END
ccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccc
      SUBROUTINE HAMIL(CRV,x)
c
c     Hamiltonian Matrix
c
      IMPLICIT NONE
      INTEGER NN
      REAL x,VPOT1
      COMPLEX CRV
      PARAMETER(NN=1)
      DIMENSION CRV(NN,NN)
c
      CALL VA(VPOT1,x)
      CRV(1,1)=VPOT1
c
      RETURN
      END
ccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccc
      SUBROUTINE VA(V,x)
c
c     Potential Energy Surface: Harmonic Oscillator
c
      implicit none
      REAL V,x,rmass,xk,pk,rk,omega
      common /packet/ rmass,xk,pk
      omega=1.0
      rk=rmass*omega**2
      V=0.5*rk*x*x
      RETURN
      END
ccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccc
      subroutine SetKinProp(dt,tprop)
c
c     Kinetic Energy part of the Trotter Expansion: exp(-i p^2 dt/(2 m))
c
      IMPLICIT NONE
      INTEGER nptx,kx,nx,npts
      REAL xsc,xmin,xmax,propfacx,rmass,xk,pi,alenx,dt,pk
      COMPLEX tprop,eye
      parameter(npts=9,nptx=2**npts)
      DIMENSION tprop(nptx)
      common /xy/  xmin,xmax
      common /packet/  rmass,xk,pk
c
```

```
      eye=(0.,1.)
      pi = acos(-1.0)
      alenx=xmax-xmin
      propfacx=-dt/2./rmass*(2.*pi)**2
      do kx=1,nptx
         if(kx.le.(nptx/2+1)) then
            nx=kx-1
         else
            nx=kx-1-nptx
         end if
         xsc=0.
         if(nx.ne.0) xsc=real(nx)/alenx
         tprop(kx)=exp(eye*(propfacx*xsc**2))
       end do
c
      return
      end
ccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccc
      subroutine SetPotProp(dt,vprop)
c
c     Potential Energy part of the Trotter Expansion: exp(-i V dt/2)
c
      IMPLICIT NONE
      INTEGER NN,ii,nptx,npts
      REAL xmin,xmax,dx,dt,x,VPOT
      COMPLEX vprop,eye
      parameter(npts=9,nptx=2**npts,NN=1)
      DIMENSION vprop(nptx,NN,NN)
      common /xy/  xmin,xmax
      eye=(0.,1.)
      dx=(xmax-xmin)/real(nptx)
c
      do ii=1,nptx
         x=xmin+ii*dx
         CALL VA(VPOT,x)
         vprop(ii,1,1)=exp(-eye*0.5*dt*VPOT)/sqrt(nptx*1.0)
      END DO
      RETURN
      END
ccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccc
      SUBROUTINE energies(energy)
      IMPLICIT NONE
      INTEGER j,NN
      COMPLEX energy,RV,RKE
      PARAMETER (NN=1)
      DIMENSION RV(NN),RKE(NN),energy(NN)
      CALL PE(RV)
      CALL KE(RKE)
      DO j=1,NN
         energy(j)=RV(j)+RKE(j)
      END DO
      RETURN
      END
ccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccc
      FUNCTION Psia(x,istep,dt)
c
c     Analytic wave-packet <x|Psia(istep)> obtained by applying the
c     harmonic propagator to the initial state,
c     <x'|Psi(0)> = (alpha/pi)**.25*exp(-alpha/2*(x'-xk)**2+eye*pk*(x'-xk)),
c     where the propagator is
c     <x|exp(-beta H)|x'> = A exp(-rgamma*(x**2+x'**2)+rgammap*x*x'), with
```

```
c     A = sqrt(m*omega/(pi*(exp(beta*omega)-exp(-beta*omega)))), beta = i*t,
c     rgamma = 0.5*m*omega*cosh(beta*omega)/sinh(beta*omega) and
c     rgammap = m*omega/sinh(beta*omega).
c
      IMPLICIT NONE
      INTEGER istep
      REAL pk,rmass,xk,dt,x,t,omega,pi,alpha
      COMPLEX eye,Psia,beta,A,rgamma,rgammap,c0,c1,c2
      common /packet/  rmass,xk,pk
      eye=(0.0,1.0)
      omega=1.0
      alpha = omega*rmass
      pi=acos(-1.0)
      beta = eye*dt*istep
      IF(abs(beta).EQ.0) beta = eye*1.0E-7
      A = sqrt(rmass*omega/(pi*(exp(beta*omega)-exp(-beta*omega))))
      rgamma=0.5*rmass*omega*(exp(beta*omega)+exp(-beta*omega))
     1     /(exp(beta*omega)-exp(-beta*omega))
      rgammap=2.*rmass*omega/(exp(beta*omega)-exp(-beta*omega))
      c0=-eye*pk*xk-alpha/2.*xk**2
      c1=rgammap*x+alpha*xk+eye*pk
      c2=rgamma+alpha/2.
c
      Psia = A*(alpha/pi)**.25*sqrt(pi/c2)*
     1     exp(-rgamma*x**2)*exp(c0+c1**2/(4.0*c2))
c
      return
      end
ccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccc
      SUBROUTINE SAVEWF(je2,ndump,dt)
c
c     Dump Time Evolved Wave packet
c
      IMPLICIT NONE
      INTEGER je2,nptx,npts,kk,NN,ncount,ndump,jj
      COMPLEX chi,CRV,energy,psi,Psia
      character*9 B
      REAL V,x1,c1,c2,c1a,x,xmin,xmax,dx,EVALUES,dt
      PARAMETER(npts=9,nptx=2**npts,NN=1)
      DIMENSION CRV(NN,NN),energy(NN),EVALUES(NN)
      DIMENSION psi(NN,NN)
      common /xy/  xmin,xmax
      COMMON / wfunc/ chi(nptx,NN)
c
      CALL energies(energy)
      jj=je2/ndump
      write(B, '(A,i4.4)') 'arch.', jj
      OPEN(1,FILE=B)
      dx=(xmax-xmin)/real(nptx)
      ncount=(je2-1)/ndump
c
c     Save Wave-packet components
c
      do kk=1,nptx
         x=xmin+kk*dx
         c1=chi(kk,1)*conjg(chi(kk,1))
         c1a=Psia(x,je2,dt)*conjg(Psia(x,je2,dt))
         write(1,33) x,sqrt(c1)+real(energy(1))
     1        ,sqrt(c1a)+real(energy(1))
      end do
      write(1,33)
```

```
      do kk=1,nptx
         x=xmin+kk*dx
         write(1,33) x
     1        ,real(chi(kk,1))+real(energy(1))
     1        ,real(Psia(x,je2,dt))+real(energy(1))
      end do
      write(1,33)
c
c     Save Adiabatic states
c
      do kk=1,nptx
         x=xmin+kk*dx
         CALL HAMIL(CRV,x)
         write(1,33) x,CRV(1,1)
      end do
      CLOSE(1)
 33   format(6(e13.6,2x))
      RETURN
      END
ccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccc
      SUBROUTINE XM(RV)
c
c     Expectation Value of the Position
c
      IMPLICIT NONE
      INTEGER nptx,npts,kk,NN,j
      COMPLEX chi,EYE,RV
      REAL Vpot,omega,xmin,xmax,dx,pi,rmass,xk,pk,x,alpha
      PARAMETER(npts=9,nptx=2**npts,NN=1)
      DIMENSION RV(NN)
      COMMON / wfunc/ chi(nptx,NN)
      common /xy/  xmin,xmax
      common /packet/rmass,xk,pk

      dx=(xmax-xmin)/real(nptx)
      DO j=1,NN
         RV(j)=0.0
         do kk=1,nptx
            x=xmin+kk*dx
            IF(j.EQ.1) CALL VA(Vpot,x)
            RV(j)=RV(j)+chi(kk,j)*x*conjg(chi(kk,j))*dx
         end do
      END DO
      RETURN
      END
ccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccc
      SUBROUTINE PE(RV)
c
c     Expectation Value of the Potential Enegy
c
      IMPLICIT NONE
      INTEGER nptx,npts,kk,NN,j
      COMPLEX chi,EYE,RV
      REAL Vpot,omega,xmin,xmax,dx,pi,rmass,xk,pk,x,alpha
      PARAMETER(npts=9,nptx=2**npts,NN=1)
      DIMENSION RV(NN)
      COMMON / wfunc/ chi(nptx,NN)
      common /xy/  xmin,xmax
      common /packet/rmass,xk,pk

      dx=(xmax-xmin)/real(nptx)
```

```
      DO j=1,NN
         RV(j)=0.0
         do kk=1,nptx
            x=xmin+kk*dx
            IF(j.EQ.1) CALL VA(Vpot,x)
            RV(j)=RV(j)+chi(kk,j)*Vpot*conjg(chi(kk,j))*dx
         end do
      END DO
      RETURN
      END
ccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccc
      subroutine KE(RKE)
c
c     Expectation value of the kinetic energy
c
      IMPLICIT NONE
      INTEGER NN,kk,nptx,kx,nx,npts,j
      REAL dp,theta,wm,p,xmin,xmax,rmass,xk,pi,alenx,pk,rm,re,ri,dx
      COMPLEX eye,chi,Psip,chic,RKE
      parameter(npts=9,nptx=2**npts,NN=1)
      DIMENSION chic(nptx),RKE(NN)
      common /xy/  xmin,xmax
      common /packet/  rmass,xk,pk
      COMMON / wfunc/ chi(nptx,NN)
c
      pi = acos(-1.0)
      dx=(xmax-xmin)/nptx
      dp=2.*pi/(xmax-xmin)
c
      DO j=1,NN
         RKE(j)=0.0
         do kk=1,nptx
            chic(kk)=chi(kk,j)
         end do
         CALL fourn(chic,nptx,1,-1)
         do kx=1,nptx
            if(kx.le.(nptx/2+1)) then
               nx=kx-1
            else
               nx=kx-1-nptx
            end if
            p=0.
            if(nx.ne.0) p = real(nx)*dp
            chic(kx)=p**2/(2.0*rmass)*chic(kx)/nptx
         end do
         CALL fourn(chic,nptx,1,1)
         do kk=1,nptx
            RKE(j)=RKE(j)+conjg(chi(kk,j))*chic(kk)*dx
         end do
      END DO
      return
      end
ccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccc
      subroutine PM(RKE)
c
c     Expectation value of the kinetic energy
c
      IMPLICIT NONE
      INTEGER NN,kk,nptx,kx,nx,npts,j
      REAL dp,theta,wm,p,xmin,xmax,rmass,xk,pi,alenx,pk,rm,re,ri,dx
      COMPLEX eye,chi,Psip,chic,RKE
```

```
      parameter(npts=9,nptx=2**npts,NN=1)
      DIMENSION chic(nptx),RKE(NN)
      common /xy/  xmin,xmax
      common /packet/  rmass,xk,pk
      COMMON / wfunc/ chi(nptx,NN)
c
      pi = acos(-1.0)
      dx=(xmax-xmin)/nptx
      dp=2.*pi/(xmax-xmin)
c
      DO j=1,NN
         RKE(j)=0.0
         do kk=1,nptx
            chic(kk)=chi(kk,j)
         end do
         CALL fourn(chic,nptx,1,-1)
         do kx=1,nptx
            if(kx.le.(nptx/2+1)) then
               nx=kx-1
            else
               nx=kx-1-nptx
            end if
            p=0.
            if(nx.ne.0) p = real(nx)*dp
            chic(kx)=p*chic(kx)/nptx
         end do
         CALL fourn(chic,nptx,1,1)
         do kk=1,nptx
            RKE(j)=RKE(j)+conjg(chi(kk,j))*chic(kk)*dx
         end do
      END DO
      return
      end
ccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccc
      SUBROUTINE PROPAGATE(vprop,tprop)
c
c     Split Operator Fourier Transform Propagation Method
c     J. Comput. Phys. 47, 412 (1982); J. Chem. Phys. 78, 301 (1983)
c
      IMPLICIT NONE
      INTEGER i,j,NN,ii,nptx,npts
      COMPLEX chi,vprop,chin1,chin2,tprop
      PARAMETER(npts=9,nptx=2**npts,NN=1)
      DIMENSION chin1(nptx),chin2(nptx)
      DIMENSION tprop(nptx),vprop(nptx,NN,NN)
      COMMON / wfunc/ chi(nptx,NN)
c
c     Apply potential energy part of the Trotter Expansion
c
      DO i=1,nptx
         chin1(i)=0.0
         DO j=1,NN
            chin1(i)=chin1(i)+vprop(i,1,j)*chi(i,j)
         END DO
      END DO
c
c     Fourier Transform wave-packet to the momentum representation
c
      CALL fourn(chin1,nptx,1,-1)
c
c     Apply kinetic energy part of the Trotter Expansion
```

```
c
      DO i=1,nptx
         chin1(i)=tprop(i)*chin1(i)
      END DO
c
c     Inverse Fourier Transform wave-packet to the coordinate representation
c
      CALL fourn(chin1,nptx,1,1)
c
c     Apply potential energy part of the Trotter Expansion
c
      DO i=1,nptx
         DO j=1,NN
            chi(i,j)=vprop(i,j,1)*chin1(i)
         END DO
      END DO
      END
ccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccc
c     Subroutine for FFT from Numerical Recipes
ccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccc
      SUBROUTINE FOURN(DATA,NN,NDIM,ISIGN)
      REAL*8 WR,WI,WPR,WPI,WTEMP,THETA
      DIMENSION NN(NDIM),DATA(*)
      NTOT=1
      DO 11 IDIM=1,NDIM
         NTOT=NTOT*NN(IDIM)
 11   CONTINUE
      NPREV=1
      DO 18 IDIM=1,NDIM
         N=NN(IDIM)
         NREM=NTOT/(N*NPREV)
         IP1=2*NPREV
         IP2=IP1*N
         IP3=IP2*NREM
         I2REV=1
         DO 14 I2=1,IP2,IP1
            IF(I2.LT.I2REV)THEN
               DO 13 I1=I2,I2+IP1-2,2
                  DO 12 I3=I1,IP3,IP2
                     I3REV=I2REV+I3-I2
                     TEMPR=DATA(I3)
                     TEMPI=DATA(I3+1)
                     DATA(I3)=DATA(I3REV)
                     DATA(I3+1)=DATA(I3REV+1)
                     DATA(I3REV)=TEMPR
                     DATA(I3REV+1)=TEMPI
 12               CONTINUE
 13            CONTINUE
            ENDIF
            IBIT=IP2/2
 1          IF ((IBIT.GE.IP1).AND.(I2REV.GT.IBIT)) THEN
               I2REV=I2REV-IBIT
               IBIT=IBIT/2
               GO TO 1
            ENDIF
            I2REV=I2REV+IBIT
 14      CONTINUE
         IFP1=IP1
 2       IF(IFP1.LT.IP2)THEN
            IFP2=2*IFP1
            THETA=ISIGN*6.28318530717959D0/(IFP2/IP1)
```

```
            WPR=-2.D0*DSIN(0.5D0*THETA)**2
            WPI=DSIN(THETA)
            WR=1.D0
            WI=0.D0
            DO 17 I3=1,IFP1,IP1
               DO 16 I1=I3,I3+IP1-2,2
                  DO 15 I2=I1,IP3,IFP2
                     K1=I2
                     K2=K1+IFP1
                     TEMPR=SNGL(WR)*DATA(K2)-SNGL(WI)*DATA(K2+1)
                     TEMPI=SNGL(WR)*DATA(K2+1)+SNGL(WI)*DATA(K2)
                     DATA(K2)=DATA(K1)-TEMPR
                     DATA(K2+1)=DATA(K1+1)-TEMPI
                     DATA(K1)=DATA(K1)+TEMPR
                     DATA(K1+1)=DATA(K1+1)+TEMPI
 15               CONTINUE
 16            CONTINUE
               WTEMP=WR
               WR=WR*WPR-WI*WPI+WR
               WI=WI*WPR+WTEMP*WPI+WI
 17         CONTINUE
            IFP1=IFP2
            GO TO 2
         ENDIF
         NPREV=N*NPREV
 18   CONTINUE
      RETURN
      END
cccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccc
```

Problem 8:

The output of this program is analogous to Problem 6 but for a Morse potential. Cut the source code attached bellow, save it in a file named Problem8.f, compile it by typing

```
f77 Problem8.f -o Problem8
```

run it by typing

```
./Problem8
```

Visualize the output of the time dependent expectation values as compared to classical trajectories as follows: type

```
gnuplot
```

then type

```
set dat sty line
```

then type

```
plot ''traj.0000''
```

That will show the numerical computation of the expectation value $< \Psi_t|\hat{x}|\Psi_t >$ as a function of time. In order to visualize the classical result on top of the quantum mechanical expectation value, type

```
replot ''traj.0000'' u 1:4
```

In order to visualize the output of $< \Psi_t|\hat{p}|\Psi_t >$ as a function of time, type

```
plot ''traj.0000'' u 1:3
```

and to visualize the classical result on top of the quantum mechanical expectation value, type

```
replot ''traj.0000'' u 1:5
```

The plot of $< \Psi_t|\hat{p}|\Psi_t >$ vs. $< \Psi_t|\hat{x}|\Psi_t >$ can be obtained by typing

```
plot ''traj.0000'' u 3:2
```

and the corresponding classical results $p(t)$ vs. $x(t)$

```
plot ''traj.0000'' u 5:4
```

To exit, type

```
quit
```

The snapshots of the time-dependent wave-packet can be visualized as a movie by typing

```
gnuplot<pp_8
```

where the file named

```
pp_8
```

has the following lines:

```
set yrange[0:9]
set xrange[-5:25]
set dat sty l
plot "arch.0001" u 1:2 lw 3
pause .1
plot "arch.0002" u 1:2 lw 3
pause .1
plot "arch.0003" u 1:2 lw 3
pause .1
plot "arch.0004" u 1:2 lw 3
pause .1
plot "arch.0005" u 1:2 lw 3
pause .1
plot "arch.0006" u 1:2 lw 3
pause .1
plot "arch.0007" u 1:2 lw 3
pause .1
plot "arch.0008" u 1:2 lw 3
pause .1
plot "arch.0009" u 1:2 lw 3
pause .1
plot "arch.0010" u 1:2 lw 3
pause .1
plot "arch.0011" u 1:2 lw 3
pause .1
plot "arch.0012" u 1:2 lw 3
pause .1
plot "arch.0013" u 1:2 lw 3
pause .1
plot "arch.0014" u 1:2 lw 3
pause .1
plot "arch.0015" u 1:2 lw 3
pause .1
plot "arch.0016" u 1:2 lw 3
pause .1
plot "arch.0017" u 1:2 lw 3
pause .1
plot "arch.0018" u 1:2 lw 3
pause .1
plot "arch.0019" u 1:2 lw 3
pause .1
plot "arch.0020" u 1:2 lw 3
pause .1
plot "arch.0021" u 1:2 lw 3
pause .1
plot "arch.0022" u 1:2 lw 3
pause .1
plot "arch.0023" u 1:2 lw 3
pause .1
plot "arch.0024" u 1:2 lw 3
pause .1
plot "arch.0025" u 1:2 lw 3
pause .1
plot "arch.0026" u 1:2 lw 3
pause .1
plot "arch.0027" u 1:2 lw 3
pause .1
plot "arch.0028" u 1:2 lw 3
pause .1
plot "arch.0029" u 1:2 lw 3
pause .1
```

```
plot "arch.0030" u 1:2 lw 3
pause .1
plot "arch.0031" u 1:2 lw 3
pause .1
plot "arch.0032" u 1:2 lw 3
pause .1
plot "arch.0033" u 1:2 lw 3
pause .1
plot "arch.0034" u 1:2 lw 3
pause .1
plot "arch.0035" u 1:2 lw 3
pause .1
plot "arch.0036" u 1:2 lw 3
pause .1
plot "arch.0037" u 1:2 lw 3
pause .1
plot "arch.0038" u 1:2 lw 3
pause .1
plot "arch.0039" u 1:2 lw 3
pause .1
plot  "arch.0040" u 1:2 lw 3
pause .1
plot "arch.0041" u 1:2 lw 3
pause .1
plot "arch.0042" u 1:2 lw 3
pause .1
plot "arch.0043" u 1:2 lw 3
pause .1
plot "arch.0044" u 1:2 lw 3
pause .1
plot "arch.0045" u 1:2 lw 3
pause .1
plot "arch.0046" u 1:2 lw 3
pause .1
plot "arch.0047" u 1:2 lw 3
pause .1
plot "arch.0048" u 1:2 lw 3
pause .1
plot "arch.0049" u 1:2 lw 3
pause .1
plot "arch.0050" u 1:2 lw 3
pause .1
plot "arch.0051" u 1:2 lw 3
pause .1
plot "arch.0052" u 1:2 lw 3
pause .1
plot "arch.0053" u 1:2 lw 3
pause .1
plot "arch.0054" u 1:2 lw 3
pause .1
plot "arch.0055" u 1:2 lw 3
pause .1
plot "arch.0056" u 1:2 lw 3
pause .1
plot "arch.0057" u 1:2 lw 3
pause .1
plot "arch.0058" u 1:2 lw 3
pause .1
plot "arch.0059" u 1:2 lw 3
pause .1
plot "arch.0060" u 1:2 lw 3
```

```
pause .1
plot "arch.0061" u 1:2 lw 3
pause .1
plot "arch.0062" u 1:2 lw 3
pause .1
plot "arch.0063" u 1:2 lw 3
pause .1
plot "arch.0064" u 1:2 lw 3
pause .1
plot "arch.0065" u 1:2 lw 3
pause .1
plot "arch.0066" u 1:2 lw 3
pause .1
plot "arch.0067" u 1:2 lw 3
pause .1
plot "arch.0068" u 1:2 lw 3
pause .1
plot "arch.0069" u 1:2 lw 3
pause .1
plot "arch.0070" u 1:2 lw 3
pause .1
plot "arch.0071" u 1:2 lw 3
pause .1
plot "arch.0072" u 1:2 lw 3
pause .1
plot "arch.0073" u 1:2 lw 3
pause .1
plot "arch.0074" u 1:2 lw 3
pause .1
plot "arch.0075" u 1:2 lw 3
pause .1
plot "arch.0076" u 1:2 lw 3
pause .1
plot "arch.0077" u 1:2 lw 3
pause .1
plot "arch.0078" u 1:2 lw 3
pause .1
plot "arch.0079" u 1:2 lw 3
pause .1
plot  "arch.0080" u 1:2 lw 3
pause .1
plot "arch.0081" u 1:2 lw 3
pause .1
plot "arch.0082" u 1:2 lw 3
pause .1
plot "arch.0083" u 1:2 lw 3
pause .1
plot "arch.0084" u 1:2 lw 3
pause .1
plot "arch.0085" u 1:2 lw 3
pause .1
plot "arch.0086" u 1:2 lw 3
pause .1
plot "arch.0087" u 1:2 lw 3
pause .1
plot "arch.0088" u 1:2 lw 3
pause .1
plot "arch.0089" u 1:2 lw 3
pause .1
plot  "arch.0090" u 1:2 lw 3
pause .1
```

```
plot "arch.0091" u 1:2 lw 3
pause .1
plot "arch.0092" u 1:2 lw 3
pause .1
plot "arch.0093" u 1:2 lw 3
pause .1
plot "arch.0094" u 1:2 lw 3
pause .1
plot "arch.0095" u 1:2 lw 3
pause .1
plot "arch.0096" u 1:2 lw 3
pause .1
plot "arch.0097" u 1:2 lw 3
pause .1
plot "arch.0098" u 1:2 lw 3
pause .1
plot "arch.0099" u 1:2 lw 3
pause .1
```

```
      PROGRAM Problem8
c
c     1-D wave packet propagation and Velocity-Verlet propagation
c     on a Morse potential energy surface
c
      IMPLICIT NONE
      INTEGER NN,npts,nptx,ndump
      INTEGER istep,nstep,jj
      REAL dt,xc,pc
      COMPLEX vprop,tprop,x_mean,p_mean
      character*9 Bfile
      PARAMETER(npts=10,nptx=2**npts,NN=1)
      DIMENSION vprop(nptx,NN,NN),tprop(nptx)
      DIMENSION x_mean(NN),p_mean(NN)
      COMMON /class/ xc,pc
c     xo
      jj=0
      write(Bfile, '(A,i4.4)') 'traj.', jj
      OPEN(10,FILE=Bfile)
      CALL ReadParam(nstep,ndump,dt)
      call Initialize()
      CALL SetKinProp(dt,tprop)
      CALL SetPotProp(dt,vprop)
      DO istep=1,nstep+1
         IF(mod(istep-1,10).EQ.0)
     1        PRINT *, "Step=", istep-1,", Final step=", nstep
         IF(istep.GE.1) CALL PROPAGATE(vprop,tprop)
         IF(mod((istep-1),ndump).EQ.0) THEN
            CALL SAVEWF(istep,ndump,dt)
            CALL XM(x_mean)
            CALL PM(p_mean)
            CALL VV(dt)
            WRITE(10,22) (istep-1.)*dt
     1           ,real(x_mean(1)),real(p_mean(1)),xc,pc
         END IF
      END DO
      CLOSE(10)
 22   FORMAT(6(e13.6,2x))
      END
ccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccc
      subroutine ReadParam(nstep,ndump,dt)
c
c     Parameters defining the grid (xmin, xmax), integration time step (dt),
c     rmass (rmass), initial position (xk), initial momentum (pk),
c     number of propagation steps (nstep), and how often to save a pic (ndump)
c
      IMPLICIT NONE
      INTEGER ntype,nstep,nrpt,ireport,ndump,nlit
      REAL xmin,xmax,pk,rmass,xk,dt
      common /packet/  rmass,xk,pk
      common /xy/  xmin,xmax
c
      xmin=-5.0
      xmax=25.0
      dt=0.2
      rmass=1.0
      xk=-.5
      pk=0.0
      nstep=100
      ndump=1
c
```

```
      return
      end
ccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccc
      SUBROUTINE VV(dt)
c
c     Velocity Verlet Algorithm J. Chem. Phys. 76, 637 (1982)
c
      IMPLICIT NONE
      REAL v,dx,dt,xc,pc,rmass,xk,pk,acc,xt,VPOT1,VPOT2,F
      COMMON /class/ xc,pc
      common /packet/  rmass,xk,pk
c
c     Compute Force
c
      dx=0.01
      xt=xc+dx
      CALL VA(VPOT1,xt)
      xt=xc-dx
      CALL VA(VPOT2,xt)
      F=-(VPOT1-VPOT2)/(2.0*dx)
      v=pc/rmass
c
c     Advance momenta half a step
c
      pc=pc+0.5*F*dt
c
c     Advance coordinates a step
c
      xc=xc+v*dt+0.5*dt**2*F/rmass
c
c     Compute Force
c
      dx=0.01
      xt=xc+dx
      CALL VA(VPOT1,xt)
      xt=xc-dx
      CALL VA(VPOT2,xt)
      F=-(VPOT1-VPOT2)/(2.0*dx)
c
c     Advance momenta half a step
c
      pc=pc+0.5*F*dt
c
      return
      end
ccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccc
      SUBROUTINE Initialize()

      IMPLICIT NONE
      INTEGER NN,nptx,npts,kk
      COMPLEX chi0,chi,EYE,CRV
      REAL xc,pc,omega,xk2,xmin,xmax,dx,pi,rmass,xk,pk,x,alpha,alpha2
      PARAMETER(npts=10,nptx=2**npts,NN=1)
      DIMENSION CRV(NN,NN)
      common /xy/  xmin,xmax
      common /packet/  rmass,xk,pk
      COMMON / wfunc/ chi(nptx,NN)
      COMMON / iwfunc/ chi0(nptx,NN)
      COMMON /class/ xc,pc

      EYE=(0.0,1.0)
```

```
      pi= acos(-1.0)
      omega=1.
      dx=(xmax-xmin)/real(nptx)
      xc=xk
      pc=pk
c
c     Wave Packet Initialization: Gaussian centered at xk, with momentum pk
c
      alpha=rmass*omega
      do kk=1,nptx
         x=xmin+kk*dx
         chi(kk,1)=((alpha/pi)**0.25)
     1        *exp(-alpha/2.*(x-xk)**2+EYE*pk*(x-xk))
         chi0(kk,1)=chi(kk,1)
      end do
      RETURN
      END
ccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccc
      SUBROUTINE HAMIL(CRV,x)
c
c     Hamiltonian Matrix
c
      IMPLICIT NONE
      INTEGER NN
      REAL x,VPOT1
      COMPLEX CRV
      PARAMETER(NN=1)
      DIMENSION CRV(NN,NN)
c
      CALL VA(VPOT1,x)
      CRV(1,1)=VPOT1
c
      RETURN
      END
ccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccc
      SUBROUTINE VA(V,x)
c
c     Potential Energy Surface: Morse Potential [Phys. Rev. (1929) 34:57]
c
      implicit none
      REAL V,x,rmass,xk,pk,rk,omega,De,xeq,a
      common /packet/ rmass,xk,pk
      xeq=0.0
      omega=1.0
      De=8.0
      rk=rmass*omega**2
      a=sqrt(rk/(2.0*De))
      V=De*(1.0-exp(-a*(x-xeq)))**2
      RETURN
      END
ccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccc
      subroutine SetKinProp(dt,tprop)
c
c     Kinetic Energy part of the Trotter Expansion: exp(-i p^2 dt/(2 m))
c
      IMPLICIT NONE
      INTEGER nptx,kx,nx,npts
      REAL xsc,xmin,xmax,propfacx,rmass,xk,pi,alenx,dt,pk
      COMPLEX tprop,eye
      parameter(npts=10,nptx=2**npts)
      DIMENSION tprop(nptx)
```

```
      common /xy/  xmin,xmax
      common /packet/  rmass,xk,pk
c
      eye=(0.,1.)
      pi = acos(-1.0)
      alenx=xmax-xmin
      propfacx=-dt/2./rmass*(2.*pi)**2
      do kx=1,nptx
         if(kx.le.(nptx/2+1)) then
            nx=kx-1
         else
            nx=kx-1-nptx
         end if
         xsc=0.
         if(nx.ne.0) xsc=real(nx)/alenx
         tprop(kx)=exp(eye*(propfacx*xsc**2))
       end do
c
      return
      end
ccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccc
      subroutine SetPotProp(dt,vprop)
c
c     Potential Energy part of the Trotter Expansion: exp(-i V dt/2)
c
      IMPLICIT NONE
      INTEGER NN,ii,nptx,npts
      REAL xmin,xmax,dx,dt,x,VPOT
      COMPLEX vprop,eye
      parameter(npts=10,nptx=2**npts,NN=1)
      DIMENSION vprop(nptx,NN,NN)
      common /xy/  xmin,xmax
      eye=(0.,1.)
      dx=(xmax-xmin)/real(nptx)
c
      do ii=1,nptx
         x=xmin+ii*dx
         CALL VA(VPOT,x)
         vprop(ii,1,1)=exp(-eye*0.5*dt*VPOT)/sqrt(nptx*1.0)
      END DO
      RETURN
      END
ccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccc
      SUBROUTINE energies(energy)
      IMPLICIT NONE
      INTEGER j,NN
      COMPLEX energy,RV,RKE
      PARAMETER (NN=1)
      DIMENSION RV(NN),RKE(NN),energy(NN)
      CALL PE(RV)
      CALL KE(RKE)
      DO j=1,NN
         energy(j)=RV(j)+RKE(j)
      END DO
      RETURN
      END
ccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccc
      SUBROUTINE SAVEWF(je2,ndump,dt)
c
c     Dump Time Evolved Wave packet
c
```

```
      IMPLICIT NONE
      INTEGER je2,nptx,npts,kk,NN,ncount,ndump,jj
      COMPLEX chi,CRV,energy,psi,Psia
      character*9 B
      REAL V,x1,c1,c2,c1a,x,xmin,xmax,dx,EVALUES,dt
      PARAMETER(npts=10,nptx=2**npts,NN=1)
      DIMENSION CRV(NN,NN),EVALUES(NN)
      DIMENSION psi(NN,NN)
      common /xy/  xmin,xmax
      COMMON / wfunc/ chi(nptx,NN)
      COMMON /ENER/ energy(NN)
c
      IF(je2.EQ.1) CALL energies(energy)
      jj=je2/ndump
      write(B, '(A,i4.4)') 'arch.', jj
      OPEN(1,FILE=B)
      dx=(xmax-xmin)/real(nptx)
      ncount=(je2-1)/ndump
c
c     Save Wave-packet components
c
      do kk=1,nptx
         x=xmin+kk*dx
         c1=chi(kk,1)*conjg(chi(kk,1))
         write(1,33) x,sqrt(c1)+real(energy(1))
      end do
      write(1,33)
      do kk=1,nptx
         x=xmin+kk*dx
         write(1,33) x,real(energy(1))
      end do
      write(1,33)
c
c     Save Adiabatic states
c
      do kk=1,nptx
         x=xmin+kk*dx
         CALL HAMIL(CRV,x)
         write(1,33) x,CRV(1,1)
      end do
      CLOSE(1)
 33   format(6(e13.6,2x))
      RETURN
      END
ccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccc
      SUBROUTINE XM(RV)
c
c     Expectation Value of the Position
c
      IMPLICIT NONE
      INTEGER nptx,npts,kk,NN,j
      COMPLEX chi,EYE,RV
      REAL Vpot,omega,xmin,xmax,dx,pi,rmass,xk,pk,x,alpha
      PARAMETER(npts=10,nptx=2**npts,NN=1)
      DIMENSION RV(NN)
      COMMON / wfunc/ chi(nptx,NN)
      common /xy/  xmin,xmax
      common /packet/rmass,xk,pk

      dx=(xmax-xmin)/real(nptx)
      DO j=1,NN
```

```
         RV(j)=0.0
         do kk=1,nptx
            x=xmin+kk*dx
            IF(j.EQ.1) CALL VA(Vpot,x)
            RV(j)=RV(j)+chi(kk,j)*x*conjg(chi(kk,j))*dx
         end do
      END DO
      RETURN
      END
ccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccc
      SUBROUTINE PE(RV)
c
c     Expectation Value of the Potential Enegy
c
      IMPLICIT NONE
      INTEGER nptx,npts,kk,NN,j
      COMPLEX chi,EYE,RV
      REAL Vpot,omega,xmin,xmax,dx,pi,rmass,xk,pk,x,alpha
      PARAMETER(npts=10,nptx=2**npts,NN=1)
      DIMENSION RV(NN)
      COMMON / wfunc/ chi(nptx,NN)
      common /xy/  xmin,xmax
      common /packet/rmass,xk,pk

      dx=(xmax-xmin)/real(nptx)
      DO j=1,NN
         RV(j)=0.0
         do kk=1,nptx
            x=xmin+kk*dx
            IF(j.EQ.1) CALL VA(Vpot,x)
            RV(j)=RV(j)+chi(kk,j)*Vpot*conjg(chi(kk,j))*dx
         end do
      END DO
      RETURN
      END
ccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccc
      subroutine KE(RKE)
c
c     Expectation value of the kinetic energy
c
      IMPLICIT NONE
      INTEGER NN,kk,nptx,kx,nx,npts,j
      REAL dp,theta,wm,p,xmin,xmax,rmass,xk,pi,alenx,pk,rm,re,ri,dx
      COMPLEX eye,chi,Psip,chic,RKE
      parameter(npts=10,nptx=2**npts,NN=1)
      DIMENSION chic(nptx),RKE(NN)
      common /xy/  xmin,xmax
      common /packet/  rmass,xk,pk
      COMMON / wfunc/ chi(nptx,NN)
c
      pi = acos(-1.0)
      dx=(xmax-xmin)/nptx
      dp=2.*pi/(xmax-xmin)
c
      DO j=1,NN
         RKE(j)=0.0
         do kk=1,nptx
            chic(kk)=chi(kk,j)
         end do
         CALL fourn(chic,nptx,1,-1)
         do kx=1,nptx
```

```
            if(kx.le.(nptx/2+1)) then
               nx=kx-1
            else
               nx=kx-1-nptx
            end if
            p=0.
            if(nx.ne.0) p = real(nx)*dp
            chic(kx)=p**2/(2.0*rmass)*chic(kx)/nptx
         end do
         CALL fourn(chic,nptx,1,1)
         do kk=1,nptx
            RKE(j)=RKE(j)+conjg(chi(kk,j))*chic(kk)*dx
         end do
      END DO
      return
      end
cccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccc
      subroutine PM(RKE)
c
c     Expectation value of the kinetic energy
c
      IMPLICIT NONE
      INTEGER NN,kk,nptx,kx,nx,npts,j
      REAL dp,theta,wm,p,xmin,xmax,rmass,xk,pi,alenx,pk,rm,re,ri,dx
      COMPLEX eye,chi,Psip,chic,RKE
      parameter(npts=10,nptx=2**npts,NN=1)
      DIMENSION chic(nptx),RKE(NN)
      common /xy/  xmin,xmax
      common /packet/  rmass,xk,pk
      COMMON / wfunc/ chi(nptx,NN)
c
      pi = acos(-1.0)
      dx=(xmax-xmin)/nptx
      dp=2.*pi/(xmax-xmin)
c
      DO j=1,NN
         RKE(j)=0.0
         do kk=1,nptx
            chic(kk)=chi(kk,j)
         end do
         CALL fourn(chic,nptx,1,-1)
         do kx=1,nptx
            if(kx.le.(nptx/2+1)) then
               nx=kx-1
            else
               nx=kx-1-nptx
            end if
            p=0.
            if(nx.ne.0) p = real(nx)*dp
            chic(kx)=p*chic(kx)/nptx
         end do
         CALL fourn(chic,nptx,1,1)
         do kk=1,nptx
            RKE(j)=RKE(j)+conjg(chi(kk,j))*chic(kk)*dx
         end do
      END DO
      return
      end
cccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccc
      SUBROUTINE PROPAGATE(vprop,tprop)
c
```

```
c     Split Operator Fourier Transform Propagation Method
c     J. Comput. Phys. 47, 412 (1982); J. Chem. Phys. 78, 301 (1983)
c
      IMPLICIT NONE
      INTEGER i,j,NN,ii,nptx,npts
      COMPLEX chi,vprop,chin1,chin2,tprop
      PARAMETER(npts=10,nptx=2**npts,NN=1)
      DIMENSION chin1(nptx),chin2(nptx)
      DIMENSION tprop(nptx),vprop(nptx,NN,NN)
      COMMON / wfunc/ chi(nptx,NN)
c
c     Apply potential energy part of the Trotter Expansion
c
      DO i=1,nptx
         chin1(i)=0.0
         DO j=1,NN
            chin1(i)=chin1(i)+vprop(i,1,j)*chi(i,j)
         END DO
      END DO
c
c     Fourier Transform wave-packet to the momentum representation
c
      CALL fourn(chin1,nptx,1,-1)
c
c     Apply kinetic energy part of the Trotter Expansion
c
      DO i=1,nptx
         chin1(i)=tprop(i)*chin1(i)
      END DO
c
c     Inverse Fourier Transform wave-packet to the coordinate representation
c
      CALL fourn(chin1,nptx,1,1)
c
c     Apply potential energy part of the Trotter Expansion
c
      DO i=1,nptx
         DO j=1,NN
            chi(i,j)=vprop(i,j,1)*chin1(i)
         END DO
      END DO
      END
ccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccc
c     Subroutine for FFT from Numerical Recipes
ccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccc
      SUBROUTINE FOURN(DATA,NN,NDIM,ISIGN)
      REAL*8 WR,WI,WPR,WPI,WTEMP,THETA
      DIMENSION NN(NDIM),DATA(*)
      NTOT=1
      DO 11 IDIM=1,NDIM
         NTOT=NTOT*NN(IDIM)
 11   CONTINUE
      NPREV=1
      DO 18 IDIM=1,NDIM
         N=NN(IDIM)
         NREM=NTOT/(N*NPREV)
         IP1=2*NPREV
         IP2=IP1*N
         IP3=IP2*NREM
         I2REV=1
         DO 14 I2=1,IP2,IP1
```

```
            IF(I2.LT.I2REV)THEN
               DO 13 I1=I2,I2+IP1-2,2
                  DO 12 I3=I1,IP3,IP2
                     I3REV=I2REV+I3-I2
                     TEMPR=DATA(I3)
                     TEMPI=DATA(I3+1)
                     DATA(I3)=DATA(I3REV)
                     DATA(I3+1)=DATA(I3REV+1)
                     DATA(I3REV)=TEMPR
                     DATA(I3REV+1)=TEMPI
 12               CONTINUE
 13            CONTINUE
               ENDIF
               IBIT=IP2/2
 1             IF ((IBIT.GE.IP1).AND.(I2REV.GT.IBIT)) THEN
                  I2REV=I2REV-IBIT
                  IBIT=IBIT/2
                  GO TO 1
               ENDIF
               I2REV=I2REV+IBIT
 14         CONTINUE
            IFP1=IP1
 2          IF(IFP1.LT.IP2)THEN
               IFP2=2*IFP1
               THETA=ISIGN*6.28318530717959D0/(IFP2/IP1)
               WPR=-2.D0*DSIN(0.5D0*THETA)**2
               WPI=DSIN(THETA)
               WR=1.D0
               WI=0.D0
               DO 17 I3=1,IFP1,IP1
                  DO 16 I1=I3,I3+IP1-2,2
                     DO 15 I2=I1,IP3,IFP2
                        K1=I2
                        K2=K1+IFP1
                        TEMPR=SNGL(WR)*DATA(K2)-SNGL(WI)*DATA(K2+1)
                        TEMPI=SNGL(WR)*DATA(K2+1)+SNGL(WI)*DATA(K2)
                        DATA(K2)=DATA(K1)-TEMPR
                        DATA(K2+1)=DATA(K1+1)-TEMPI
                        DATA(K1)=DATA(K1)+TEMPR
                        DATA(K1+1)=DATA(K1+1)+TEMPI
 15                  CONTINUE
 16               CONTINUE
                  WTEMP=WR
                  WR=WR*WPR-WI*WPI+WR
                  WI=WI*WPR+WTEMP*WPI+WI
 17            CONTINUE
               IFP1=IFP2
               GO TO 2
            ENDIF
            NPREV=N*NPREV
 18      CONTINUE
         RETURN
         END
cccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccc
```

Problem 9:

The output of this program can be generated and visualized as follows. Cut the source code attached bellow, save it in a file named Problem9.f, compile it by typing

```
f77 Problem9.f -o Problem9
```

run it by typing

```
./Problem9
```

The snapshots of the time-dependent wave-packet can be visualized as a movie by typing

```
gnuplot<pp_9
```

where the file named

```
pp_9
```

has the following lines:

```
set yrange[0:4]
set xrange[-10:10]
set dat sty l
plot "arch.0001" u 1:2 lw 3
pause .1
plot "arch.0002" u 1:2 lw 3
pause .1
plot "arch.0003" u 1:2 lw 3
pause .1
plot "arch.0004" u 1:2 lw 3
pause .1
plot "arch.0005" u 1:2 lw 3
pause .1
plot "arch.0006" u 1:2 lw 3
pause .1
plot "arch.0007" u 1:2 lw 3
pause .1
plot "arch.0008" u 1:2 lw 3
pause .1
plot "arch.0009" u 1:2 lw 3
pause .1
plot "arch.0010" u 1:2 lw 3
pause .1
plot "arch.0011" u 1:2 lw 3
pause .1
plot "arch.0012" u 1:2 lw 3
pause .1
plot "arch.0013" u 1:2 lw 3
pause .1
plot "arch.0014" u 1:2 lw 3
pause .1
plot "arch.0015" u 1:2 lw 3
pause .1
plot "arch.0016" u 1:2 lw 3
pause .1
plot "arch.0017" u 1:2 lw 3
pause .1
plot "arch.0018" u 1:2 lw 3
pause .1
plot "arch.0019" u 1:2 lw 3
```

```
pause .1
plot "arch.0020" u 1:2 lw 3
pause .1
plot "arch.0021" u 1:2 lw 3
pause .1
plot "arch.0022" u 1:2 lw 3
pause .1
plot "arch.0023" u 1:2 lw 3
pause .1
plot "arch.0024" u 1:2 lw 3
pause .1
plot "arch.0025" u 1:2 lw 3
pause .1
plot "arch.0026" u 1:2 lw 3
pause .1
plot "arch.0027" u 1:2 lw 3
pause .1
plot "arch.0028" u 1:2 lw 3
pause .1
plot "arch.0029" u 1:2 lw 3
pause .1
plot "arch.0030" u 1:2 lw 3
pause .1
plot "arch.0031" u 1:2 lw 3
pause .1
plot "arch.0032" u 1:2 lw 3
pause .1
plot "arch.0033" u 1:2 lw 3
pause .1
plot "arch.0034" u 1:2 lw 3
pause .1
plot "arch.0035" u 1:2 lw 3
pause .1
plot "arch.0036" u 1:2 lw 3
pause .1
plot "arch.0037" u 1:2 lw 3
pause .1
plot "arch.0038" u 1:2 lw 3
pause .1
plot "arch.0039" u 1:2 lw 3
pause .1
plot  "arch.0040" u 1:2 lw 3
pause .1
plot "arch.0041" u 1:2 lw 3
pause .1
plot "arch.0042" u 1:2 lw 3
pause .1
plot "arch.0043" u 1:2 lw 3
pause .1
plot "arch.0044" u 1:2 lw 3
pause .1
plot "arch.0045" u 1:2 lw 3
pause .1
plot "arch.0046" u 1:2 lw 3
pause .1
plot "arch.0047" u 1:2 lw 3
pause .1
plot "arch.0048" u 1:2 lw 3
pause .1
plot "arch.0049" u 1:2 lw 3
pause .1
```

```
plot "arch.0050" u 1:2 lw 3
pause .1
plot "arch.0051" u 1:2 lw 3
pause .1
plot "arch.0052" u 1:2 lw 3
pause .1
plot "arch.0053" u 1:2 lw 3
pause .1
plot "arch.0054" u 1:2 lw 3
pause .1
plot "arch.0055" u 1:2 lw 3
pause .1
plot "arch.0056" u 1:2 lw 3
pause .1
plot "arch.0057" u 1:2 lw 3
pause .1
plot "arch.0058" u 1:2 lw 3
pause .1
plot "arch.0059" u 1:2 lw 3
pause .1
plot "arch.0060" u 1:2 lw 3
pause .1
plot "arch.0061" u 1:2 lw 3
pause .1
plot "arch.0062" u 1:2 lw 3
pause .1
plot "arch.0063" u 1:2 lw 3
pause .1
plot "arch.0064" u 1:2 lw 3
pause .1
plot "arch.0065" u 1:2 lw 3
pause .1
plot "arch.0066" u 1:2 lw 3
pause .1
plot "arch.0067" u 1:2 lw 3
pause .1
plot "arch.0068" u 1:2 lw 3
pause .1
plot "arch.0069" u 1:2 lw 3
pause .1
plot "arch.0070" u 1:2 lw 3
pause .1
plot "arch.0071" u 1:2 lw 3
pause .1
plot "arch.0072" u 1:2 lw 3
pause .1
plot "arch.0073" u 1:2 lw 3
pause .1
plot "arch.0074" u 1:2 lw 3
pause .1
plot "arch.0075" u 1:2 lw 3
pause .1
plot "arch.0076" u 1:2 lw 3
pause .1
plot "arch.0077" u 1:2 lw 3
pause .1
plot "arch.0078" u 1:2 lw 3
pause .1
plot "arch.0079" u 1:2 lw 3
pause .1
plot  "arch.0080" u 1:2 lw 3
```

```
pause .1
plot "arch.0081" u 1:2 lw 3
pause .1
plot "arch.0082" u 1:2 lw 3
pause .1
plot "arch.0083" u 1:2 lw 3
pause .1
plot "arch.0084" u 1:2 lw 3
pause .1
plot "arch.0085" u 1:2 lw 3
pause .1
plot "arch.0086" u 1:2 lw 3
pause .1
plot "arch.0087" u 1:2 lw 3
pause .1
plot "arch.0088" u 1:2 lw 3
pause .1
plot "arch.0089" u 1:2 lw 3
pause .1
plot  "arch.0090" u 1:2 lw 3
pause .1
plot "arch.0091" u 1:2 lw 3
pause .1
plot "arch.0092" u 1:2 lw 3
pause .1
plot "arch.0093" u 1:2 lw 3
pause .1
plot "arch.0094" u 1:2 lw 3
pause .1
plot "arch.0095" u 1:2 lw 3
pause .1
plot "arch.0096" u 1:2 lw 3
pause .1
plot "arch.0097" u 1:2 lw 3
pause .1
plot "arch.0098" u 1:2 lw 3
pause .1
plot "arch.0099" u 1:2 lw 3
pause .1
```

```
      PROGRAM Problem9
c
c     1-D wave packet propagation of tunneling through a barrier
c
      IMPLICIT NONE
      INTEGER NN,npts,nptx,ndump
      INTEGER istep,nstep,jj
      REAL dt,xc,pc
      COMPLEX vprop,tprop,x_mean,p_mean
      PARAMETER(npts=10,nptx=2**npts,NN=1)
      DIMENSION vprop(nptx,NN,NN),tprop(nptx)
      DIMENSION x_mean(NN),p_mean(NN)
      COMMON /class/ xc,pc
c
      CALL ReadParam(nstep,ndump,dt)
      call Initialize()
      CALL SetKinProp(dt,tprop)
      CALL SetPotProp(dt,vprop)
      DO istep=1,nstep+1
         IF(mod(istep-1,10).EQ.0)
     1        PRINT *, "Step=", istep-1,", Final step=", nstep
         IF(istep.GE.1) CALL PROPAGATE(vprop,tprop)
         IF(mod((istep-1),ndump).EQ.0) THEN
            CALL SAVEWF(istep,ndump,dt)
         END IF
      END DO
      END
ccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccc
      subroutine ReadParam(nstep,ndump,dt)
c
c     Parameters defining the grid (xmin, xmax), integration time step (dt),
c     rmass (rmass), initial position (xk), initial momentum (pk),
c     number of propagation steps (nstep), and how often to save a pic (ndump)
c
      IMPLICIT NONE
      INTEGER ntype,nstep,nrpt,ireport,ndump,nlit
      REAL xmin,xmax,pk,rmass,xk,dt
      common /packet/  rmass,xk,pk
      common /xy/  xmin,xmax
c
      xmin=-13.0
      xmax=13.0
      dt=0.1
      rmass=1.0
      xk=-4.5
      pk=1.
      nstep=100
      ndump=1
c
      return
      end
ccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccc
      SUBROUTINE Initialize()

      IMPLICIT NONE
      INTEGER NN,nptx,npts,kk
      COMPLEX chi0,chi,EYE,CRV
      REAL xc,pc,omega,xk2,xmin,xmax,dx,pi,rmass,xk,pk,x,alpha,alpha2
      PARAMETER(npts=10,nptx=2**npts,NN=1)
      DIMENSION CRV(NN,NN)
      common /xy/  xmin,xmax
```

```
      common /packet/  rmass,xk,pk
      COMMON / wfunc/ chi(nptx,NN)
      COMMON / iwfunc/ chi0(nptx,NN)
      COMMON /class/ xc,pc

      EYE=(0.0,1.0)
      pi= acos(-1.0)
      omega=1.
      dx=(xmax-xmin)/real(nptx)
      xc=xk
      pc=pk
c
c     Wave Packet Initialization: Gaussian centered at xk, with momentum pk
c
      alpha=rmass*omega
      do kk=1,nptx
         x=xmin+kk*dx
         chi(kk,1)=((alpha/pi)**0.25)
     1        *exp(-alpha/2.*(x-xk)**2+EYE*pk*(x-xk))
         chi0(kk,1)=chi(kk,1)
      end do
      RETURN
      END
ccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccc
      SUBROUTINE HAMIL(CRV,x)
c
c     Hamiltonian Matrix
c
      IMPLICIT NONE
      INTEGER NN
      REAL x,VPOT1
      COMPLEX CRV
      PARAMETER(NN=1)
      DIMENSION CRV(NN,NN)
c
      CALL VA(VPOT1,x)
      CRV(1,1)=VPOT1
c
      RETURN
      END
ccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccc
      SUBROUTINE VA(V,x)
c
c     Potential Energy Surface: Barrier
c
      implicit none
      REAL V,x,rmass,xk,pk,rk,omega
      common /packet/ rmass,xk,pk
      V=0.0
      IF(abs(x).LE.(.5)) V=3.
      RETURN
      END
ccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccc
      subroutine SetKinProp(dt,tprop)
c
c     Kinetic Energy part of the Trotter Expansion: exp(-i p^2 dt/(2 m))
c
      IMPLICIT NONE
      INTEGER nptx,kx,nx,npts
      REAL xsc,xmin,xmax,propfacx,rmass,xk,pi,alenx,dt,pk
      COMPLEX tprop,eye
```

```
      parameter(npts=10,nptx=2**npts)
      DIMENSION tprop(nptx)
      common /xy/  xmin,xmax
      common /packet/  rmass,xk,pk
c
      eye=(0.,1.)
      pi = acos(-1.0)
      alenx=xmax-xmin
      propfacx=-dt/2./rmass*(2.*pi)**2
      do kx=1,nptx
         if(kx.le.(nptx/2+1)) then
            nx=kx-1
         else
            nx=kx-1-nptx
         end if
         xsc=0.
         if(nx.ne.0) xsc=real(nx)/alenx
         tprop(kx)=exp(eye*(propfacx*xsc**2))
       end do
c
      return
      end
ccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccc
      subroutine SetPotProp(dt,vprop)
c
c     Potential Energy part of the Trotter Expansion: exp(-i V dt/2)
c
      IMPLICIT NONE
      INTEGER NN,ii,nptx,npts
      REAL xmin,xmax,dx,dt,x,VPOT,xa
      COMPLEX vprop,eye
      parameter(npts=10,nptx=2**npts,NN=1,xa=10.)
      DIMENSION vprop(nptx,NN,NN)
      common /xy/  xmin,xmax
      eye=(0.,1.)
      dx=(xmax-xmin)/real(nptx)
c
      do ii=1,nptx
         x=xmin+ii*dx
         CALL VA(VPOT,x)
         vprop(ii,1,1)=exp(-eye*0.5*dt*VPOT)/sqrt(nptx*1.0)
         IF(abs(x).GT.(xa))
     1        vprop(ii,1,1)=vprop(ii,1,1)*exp(-(abs(x)-xa)**4)
      END DO
      RETURN
      END
ccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccc
      SUBROUTINE energies(energy)
      IMPLICIT NONE
      INTEGER j,NN
      COMPLEX energy,RV,RKE
      PARAMETER (NN=1)
      DIMENSION RV(NN),RKE(NN),energy(NN)
      CALL PE(RV)
      CALL KE(RKE)
      DO j=1,NN
         energy(j)=RV(j)+RKE(j)
      END DO
      RETURN
      END
ccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccc
```

```
      SUBROUTINE SAVEWF(je2,ndump,dt)
c
c     Dump Time Evolved Wave packet
c
      IMPLICIT NONE
      INTEGER je2,nptx,npts,kk,NN,ncount,ndump,jj
      COMPLEX chi,CRV,energy,psi,Psia
      character*9 B
      REAL V,x1,c1,c2,c1a,x,xmin,xmax,dx,EVALUES,dt
      PARAMETER(npts=10,nptx=2**npts,NN=1)
      DIMENSION CRV(NN,NN),EVALUES(NN)
      DIMENSION psi(NN,NN)
      common /xy/  xmin,xmax
      COMMON / wfunc/ chi(nptx,NN)
      COMMON /ENER/ energy(NN)
c
      IF(je2.EQ.1) CALL energies(energy)
      jj=je2/ndump
      write(B, '(A,i4.4)') 'arch.', jj
      OPEN(1,FILE=B)
      dx=(xmax-xmin)/real(nptx)
      ncount=(je2-1)/ndump
c
c     Save Wave-packet components
c
      do kk=1,nptx
         x=xmin+kk*dx
         c1=chi(kk,1)*conjg(chi(kk,1))
         write(1,33) x,sqrt(c1)+real(energy(1))
      end do
      write(1,33)
      do kk=1,nptx
         x=xmin+kk*dx
         write(1,33) x
     1        ,real(chi(kk,1))+real(energy(1))
      end do
      write(1,33)
c
c     Save Adiabatic states
c
      do kk=1,nptx
         x=xmin+kk*dx
         CALL HAMIL(CRV,x)
         write(1,33) x,CRV(1,1)
      end do
      CLOSE(1)
 33   format(6(e13.6,2x))
      RETURN
      END
ccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccc
      SUBROUTINE PE(RV)
c
c     Expectation Value of the Potential Enegy
c
      IMPLICIT NONE
      INTEGER nptx,npts,kk,NN,j
      COMPLEX chi,EYE,RV
      REAL Vpot,omega,xmin,xmax,dx,pi,rmass,xk,pk,x,alpha
      PARAMETER(npts=10,nptx=2**npts,NN=1)
      DIMENSION RV(NN)
      COMMON / wfunc/ chi(nptx,NN)
```

```
      common /xy/  xmin,xmax
      common /packet/rmass,xk,pk

      dx=(xmax-xmin)/real(nptx)
      DO j=1,NN
         RV(j)=0.0
         do kk=1,nptx
            x=xmin+kk*dx
            IF(j.EQ.1) CALL VA(Vpot,x)
            RV(j)=RV(j)+chi(kk,j)*Vpot*conjg(chi(kk,j))*dx
         end do
      END DO
      RETURN
      END
ccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccc
      subroutine KE(RKE)
c
c     Expectation value of the kinetic energy
c
      IMPLICIT NONE
      INTEGER NN,kk,nptx,kx,nx,npts,j
      REAL dp,theta,wm,p,xmin,xmax,rmass,xk,pi,alenx,pk,rm,re,ri,dx
      COMPLEX eye,chi,Psip,chic,RKE
      parameter(npts=10,nptx=2**npts,NN=1)
      DIMENSION chic(nptx),RKE(NN)
      common /xy/  xmin,xmax
      common /packet/  rmass,xk,pk
      COMMON / wfunc/ chi(nptx,NN)
c
      pi = acos(-1.0)
      dx=(xmax-xmin)/nptx
      dp=2.*pi/(xmax-xmin)
c
      DO j=1,NN
         RKE(j)=0.0
         do kk=1,nptx
            chic(kk)=chi(kk,j)
         end do
         CALL fourn(chic,nptx,1,-1)
         do kx=1,nptx
            if(kx.le.(nptx/2+1)) then
               nx=kx-1
            else
               nx=kx-1-nptx
            end if
            p=0.
            if(nx.ne.0) p = real(nx)*dp
            chic(kx)=p**2/(2.0*rmass)*chic(kx)/nptx
         end do
         CALL fourn(chic,nptx,1,1)
         do kk=1,nptx
            RKE(j)=RKE(j)+conjg(chi(kk,j))*chic(kk)*dx
         end do
      END DO
      return
      end
ccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccc
      SUBROUTINE PROPAGATE(vprop,tprop)
c
c     Split Operator Fourier Transform Propagation Method
c     J. Comput. Phys. 47, 412 (1982); J. Chem. Phys. 78, 301 (1983)
```

```
c
      IMPLICIT NONE
      INTEGER i,j,NN,ii,nptx,npts
      COMPLEX chi,vprop,chin1,chin2,tprop
      PARAMETER(npts=10,nptx=2**npts,NN=1)
      DIMENSION chin1(nptx),chin2(nptx)
      DIMENSION tprop(nptx),vprop(nptx,NN,NN)
      COMMON / wfunc/ chi(nptx,NN)
c
c     Apply potential energy part of the Trotter Expansion
c
      DO i=1,nptx
         chin1(i)=0.0
         DO j=1,NN
            chin1(i)=chin1(i)+vprop(i,1,j)*chi(i,j)
         END DO
      END DO
c
c     Fourier Transform wave-packet to the momentum representation
c
      CALL fourn(chin1,nptx,1,-1)
c
c     Apply kinetic energy part of the Trotter Expansion
c
      DO i=1,nptx
         chin1(i)=tprop(i)*chin1(i)
      END DO
c
c     Inverse Fourier Transform wave-packet to the coordinate representation
c
      CALL fourn(chin1,nptx,1,1)
c
c     Apply potential energy part of the Trotter Expansion
c
      DO i=1,nptx
         DO j=1,NN
            chi(i,j)=vprop(i,j,1)*chin1(i)
         END DO
      END DO
      END
ccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccc
c     Subroutine for FFT from Numerical Recipes
ccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccc
      SUBROUTINE FOURN(DATA,NN,NDIM,ISIGN)
      REAL*8 WR,WI,WPR,WPI,WTEMP,THETA
      DIMENSION NN(NDIM),DATA(*)
      NTOT=1
      DO 11 IDIM=1,NDIM
         NTOT=NTOT*NN(IDIM)
 11   CONTINUE
      NPREV=1
      DO 18 IDIM=1,NDIM
         N=NN(IDIM)
         NREM=NTOT/(N*NPREV)
         IP1=2*NPREV
         IP2=IP1*N
         IP3=IP2*NREM
         I2REV=1
         DO 14 I2=1,IP2,IP1
            IF(I2.LT.I2REV)THEN
               DO 13 I1=I2,I2+IP1-2,2
```

```
                  DO 12 I3=I1,IP3,IP2
                     I3REV=I2REV+I3-I2
                     TEMPR=DATA(I3)
                     TEMPI=DATA(I3+1)
                     DATA(I3)=DATA(I3REV)
                     DATA(I3+1)=DATA(I3REV+1)
                     DATA(I3REV)=TEMPR
                     DATA(I3REV+1)=TEMPI
 12               CONTINUE
 13            CONTINUE
            ENDIF
            IBIT=IP2/2
 1          IF ((IBIT.GE.IP1).AND.(I2REV.GT.IBIT)) THEN
               I2REV=I2REV-IBIT
               IBIT=IBIT/2
               GO TO 1
            ENDIF
            I2REV=I2REV+IBIT
 14      CONTINUE
         IFP1=IP1
 2       IF(IFP1.LT.IP2)THEN
            IFP2=2*IFP1
            THETA=ISIGN*6.28318530717959D0/(IFP2/IP1)
            WPR=-2.D0*DSIN(0.5D0*THETA)**2
            WPI=DSIN(THETA)
            WR=1.D0
            WI=0.D0
            DO 17 I3=1,IFP1,IP1
               DO 16 I1=I3,I3+IP1-2,2
                  DO 15 I2=I1,IP3,IFP2
                     K1=I2
                     K2=K1+IFP1
                     TEMPR=SNGL(WR)*DATA(K2)-SNGL(WI)*DATA(K2+1)
                     TEMPI=SNGL(WR)*DATA(K2+1)+SNGL(WI)*DATA(K2)
                     DATA(K2)=DATA(K1)-TEMPR
                     DATA(K2+1)=DATA(K1+1)-TEMPI
                     DATA(K1)=DATA(K1)+TEMPR
                     DATA(K1+1)=DATA(K1+1)+TEMPI
 15               CONTINUE
 16            CONTINUE
               WTEMP=WR
               WR=WR*WPR-WI*WPI+WR
               WI=WI*WPR+WTEMP*WPI+WI
 17         CONTINUE
            IFP1=IFP2
            GO TO 2
         ENDIF
         NPREV=N*NPREV
 18   CONTINUE
      RETURN
      END
cccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccc
```

Problem 10:

In order to derive Eq. (28) we need to prove the following equation:

$$e^{-iV_0\tau}e^{-iV_c2\tau}e^{-iV_0\tau} = \begin{pmatrix} e^{-iV_1(\mathbf{x})2\tau}\cos(2V_c(\mathbf{x})\tau) & -i\,\sin(2V_c(\mathbf{x})\tau)\; e^{-i(\hat{V}_1(\mathbf{x})+\hat{V}_2(\mathbf{x}))\tau} \\ -i\,\sin(2V_c(\mathbf{x})\tau)\; e^{-i(V_1(\mathbf{x})+\hat{V}_2(\mathbf{x}))\tau} & \cos(2V_c(\mathbf{x})\tau)\; e^{-iV_2(\mathbf{x})2\tau} \end{pmatrix}. \tag{7}$$

where

$$e^{-iV_0\tau} = e^{-i\begin{pmatrix} V_1(\mathbf{x}) & 0 \\ 0 & V_2(\mathbf{x}) \end{pmatrix}\tau}. \tag{8}$$

Expanding the exponential on the r.h.s. of Eq. (8) gives

$$e^{-i\tau\begin{pmatrix} V_1(\mathbf{x}) & 0 \\ 0 & V_2(\mathbf{x}) \end{pmatrix}} = \begin{pmatrix} 1 & 0 \\ 0 & 1 \end{pmatrix} + \begin{pmatrix} -i\tau V_1(\mathbf{x}) & 0 \\ 0 & -i\tau V_2(\mathbf{x}) \end{pmatrix} + \begin{pmatrix} \frac{1}{2!}V_1(\mathbf{x})^2(-i\tau)^2 & 0 \\ 0 & \frac{1}{2!}V_2(\mathbf{x})^2(-i\tau)^2 \end{pmatrix} + ... \tag{9}$$

Therefore,

$$e^{-i\tau\begin{pmatrix} V_1(\mathbf{x}) & 0 \\ 0 & V_2(\mathbf{x}) \end{pmatrix}} = \begin{pmatrix} e^{-iV_1(\mathbf{x})\tau} & 0 \\ 0 & e^{-iV_2(\mathbf{x})\tau} \end{pmatrix}. \tag{10}$$

In addition, according to Eq. (30),

$$e^{-i\mathbf{V}_c2\tau} = \mathbf{D}^\dagger \begin{pmatrix} e^{iV_c(\mathbf{x})2\tau} & 0 \\ 0 & e^{-iV_c(\mathbf{x})2\tau} \end{pmatrix} \mathbf{D}, \tag{11}$$

with

$$\mathbf{D} = \mathbf{D}^\dagger \equiv \begin{pmatrix} -1/\sqrt{2} & 1/\sqrt{2} \\ 1/\sqrt{2} & 1/\sqrt{2} \end{pmatrix}, \tag{12}$$

Therefore,

$$e^{-iV_0\tau}e^{-iV_c2\tau}e^{-iV_0\tau} = \begin{pmatrix} e^{-iV_1(\mathbf{x})\tau} & 0 \\ 0 & e^{-iV_2(\mathbf{x})\tau} \end{pmatrix} \mathbf{D}^\dagger \begin{pmatrix} e^{iV_c(\mathbf{x})2\tau} & 0 \\ 0 & e^{-iV_c(\mathbf{x})2\tau} \end{pmatrix} \mathbf{D} \begin{pmatrix} e^{-iV_1(\mathbf{x})\tau} & 0 \\ 0 & e^{-iV_2(\mathbf{x})\tau} \end{pmatrix}. \tag{13}$$

The multiplication of the five matrices on the r.h.s. of Eq. (13) gives the matrix on the r.h.s. of Eq.(7).

Problem 11:

In the basis of the adiabatic states $|\phi\rangle = \mathbf{L}|\chi\rangle$, the potential energy matrix is diagonal

$$V_a = \langle\phi|\hat{V}|\phi\rangle = \begin{pmatrix} E_1(x) & 0 \\ 0 & E_2(x) \end{pmatrix}, \tag{14}$$

where $\hat{V} = E_1(x)|\phi_1\rangle + E_2(x)|\phi_2\rangle$. Substituting this diagonal matrix in the r.h.s. of Eq. (35) gives,

$$e^{-i\mathbf{V}_a 2\tau} = \begin{pmatrix} 1 & 0 \\ 0 & 1 \end{pmatrix} + \begin{pmatrix} -i\tau E_1(\mathbf{x}) & 0 \\ 0 & -i\tau E_2(\mathbf{x}) \end{pmatrix} + \begin{pmatrix} \frac{1}{2!}E_1(\mathbf{x})^2(-i\tau)^2 & 0 \\ 0 & \frac{1}{2!}E_2(\mathbf{x})^2(-i\tau)^2 \end{pmatrix} + ... \tag{15}$$

Therefore,

$$e^{-i\mathbf{V}_a 2\tau} = \begin{pmatrix} e^{-iE_1(x)2\tau} & 0 \\ 0 & e^{-iE_2(x)2\tau} \end{pmatrix} = e^{-iE2\tau}. \tag{16}$$

In addition, since the potential energy matrix has the same eigenvalues, when written in the diabatic or adiabatic representations,

$$e^{-iV_a 2\tau}|\phi\rangle = e^{-iE2\tau}|\phi\rangle, \tag{17}$$

and

$$e^{-iV_d 2\tau}|\chi\rangle = e^{-iE2\tau}|\chi\rangle = e^{-iE2\tau}\mathbf{L}^{-1}|\phi\rangle. \tag{18}$$

Therefore,

$$\mathbf{L}e^{-iV_d 2\tau}|\chi\rangle = e^{-iE2\tau}|\phi\rangle, \tag{19}$$

which gives

$$\mathbf{L}e^{-iV_d 2\tau}\mathbf{L}^{-1}|\phi\rangle = e^{-iV_a 2\tau}|\phi\rangle, \tag{20}$$

or

$$e^{-iV_d 2\tau} = \mathbf{L}^{-1}e^{-iV_a 2\tau}\mathbf{L}. \tag{21}$$

Problem 12:

The output of this program can be generated and visualized as follows. Cut the source code attached bellow, save it in a file named Problem12.f, compile it by typing

```
f77 Problem12.f -o Problem12
```

run it by typing

```
./Problem12
```

That will produce the output for item (a). In order to obtain the output for item (b), modify subroutine Hamil, so that CRV(1,2)=0.0 and CRV(2,1)=0.0, recompile and run.

The snapshots of the time-dependent wave-packet can be visualized as a movie by typing

```
gnuplot<pp_12
```

where the file named

```
pp_12
```

has the following lines:

```
set yrange[-2:5]
set dat sty l
plot "arch.0001" u 1:2 lw 3
pause .1
plot "arch.0002" u 1:2 lw 3
pause .1
plot "arch.0003" u 1:2 lw 3
pause .1
plot "arch.0004" u 1:2 lw 3
pause .1
plot "arch.0005" u 1:2 lw 3
pause .1
plot "arch.0006" u 1:2 lw 3
pause .1
plot "arch.0007" u 1:2 lw 3
pause .1
plot "arch.0008" u 1:2 lw 3
pause .1
plot "arch.0009" u 1:2 lw 3
pause .1
plot "arch.0010" u 1:2 lw 3
pause .1
plot "arch.0011" u 1:2 lw 3
pause .1
plot "arch.0012" u 1:2 lw 3
pause .1
plot "arch.0013" u 1:2 lw 3
pause .1
plot "arch.0014" u 1:2 lw 3
pause .1
plot "arch.0015" u 1:2 lw 3
pause .1
plot "arch.0016" u 1:2 lw 3
pause .1
plot "arch.0017" u 1:2 lw 3
pause .1
plot "arch.0018" u 1:2 lw 3
pause .1
```

```
plot "arch.0019" u 1:2 lw 3
pause .1
plot "arch.0020" u 1:2 lw 3
pause .1
plot "arch.0021" u 1:2 lw 3
pause .1
plot "arch.0022" u 1:2 lw 3
pause .1
plot "arch.0023" u 1:2 lw 3
pause .1
plot "arch.0024" u 1:2 lw 3
pause .1
plot "arch.0025" u 1:2 lw 3
pause .1
plot "arch.0026" u 1:2 lw 3
pause .1
plot "arch.0027" u 1:2 lw 3
pause .1
plot "arch.0028" u 1:2 lw 3
pause .1
plot "arch.0029" u 1:2 lw 3
pause .1
plot "arch.0030" u 1:2 lw 3
pause .1
plot "arch.0031" u 1:2 lw 3
pause .1
plot "arch.0032" u 1:2 lw 3
pause .1
plot "arch.0033" u 1:2 lw 3
pause .1
plot "arch.0034" u 1:2 lw 3
pause .1
plot "arch.0035" u 1:2 lw 3
pause .1
plot "arch.0036" u 1:2 lw 3
pause .1
plot "arch.0037" u 1:2 lw 3
pause .1
plot "arch.0038" u 1:2 lw 3
pause .1
plot "arch.0039" u 1:2 lw 3
pause .1
plot  "arch.0040" u 1:2 lw 3
pause .1
plot "arch.0041" u 1:2 lw 3
pause .1
plot "arch.0042" u 1:2 lw 3
pause .1
plot "arch.0043" u 1:2 lw 3
pause .1
plot "arch.0044" u 1:2 lw 3
pause .1
plot "arch.0045" u 1:2 lw 3
pause .1
plot "arch.0046" u 1:2 lw 3
pause .1
plot "arch.0047" u 1:2 lw 3
pause .1
plot "arch.0048" u 1:2 lw 3
pause .1
plot "arch.0049" u 1:2 lw 3
```

```
pause .1
plot "arch.0050" u 1:2 lw 3
pause .1
plot "arch.0051" u 1:2 lw 3
pause .1
plot "arch.0052" u 1:2 lw 3
pause .1
plot "arch.0053" u 1:2 lw 3
pause .1
plot "arch.0054" u 1:2 lw 3
pause .1
plot "arch.0055" u 1:2 lw 3
pause .1
plot "arch.0056" u 1:2 lw 3
pause .1
plot "arch.0057" u 1:2 lw 3
pause .1
plot "arch.0058" u 1:2 lw 3
pause .1
plot "arch.0059" u 1:2 lw 3
pause .1
plot "arch.0060" u 1:2 lw 3
pause .1
plot "arch.0061" u 1:2 lw 3
pause .1
plot "arch.0062" u 1:2 lw 3
pause .1
plot "arch.0063" u 1:2 lw 3
pause .1
plot "arch.0064" u 1:2 lw 3
pause .1
plot "arch.0065" u 1:2 lw 3
pause .1
plot "arch.0066" u 1:2 lw 3
pause .1
plot "arch.0067" u 1:2 lw 3
pause .1
plot "arch.0068" u 1:2 lw 3
pause .1
plot "arch.0069" u 1:2 lw 3
pause .1
plot "arch.0070" u 1:2 lw 3
pause .1
plot "arch.0071" u 1:2 lw 3
pause .1
plot "arch.0072" u 1:2 lw 3
pause .1
plot "arch.0073" u 1:2 lw 3
pause .1
plot "arch.0074" u 1:2 lw 3
pause .1
plot "arch.0075" u 1:2 lw 3
pause .1
plot "arch.0076" u 1:2 lw 3
pause .1
plot "arch.0077" u 1:2 lw 3
pause .1
plot "arch.0078" u 1:2 lw 3
pause .1
plot "arch.0079" u 1:2 lw 3
pause .1
```

```
plot  "arch.0080" u 1:2 lw 3
pause .1
plot "arch.0081" u 1:2 lw 3
pause .1
plot "arch.0082" u 1:2 lw 3
pause .1
plot "arch.0083" u 1:2 lw 3
pause .1
plot "arch.0084" u 1:2 lw 3
pause .1
plot "arch.0085" u 1:2 lw 3
pause .1
plot "arch.0086" u 1:2 lw 3
pause .1
plot "arch.0087" u 1:2 lw 3
pause .1
plot "arch.0088" u 1:2 lw 3
pause .1
plot "arch.0089" u 1:2 lw 3
pause .1
plot  "arch.0090" u 1:2 lw 3
pause .1
plot "arch.0091" u 1:2 lw 3
pause .1
plot "arch.0092" u 1:2 lw 3
pause .1
plot "arch.0093" u 1:2 lw 3
pause .1
plot "arch.0094" u 1:2 lw 3
pause .1
plot "arch.0095" u 1:2 lw 3
pause .1
plot "arch.0096" u 1:2 lw 3
pause .1
plot "arch.0097" u 1:2 lw 3
pause .1
plot "arch.0098" u 1:2 lw 3
pause .1
plot "arch.0099" u 1:2 lw 3
pause .1
```

```
      PROGRAM Problem12
c
c     1-D nonadiabatic wave-packet propagation
c
      IMPLICIT NONE
      INTEGER NN,npts,nptx,ndump
      INTEGER istep,nstep
      REAL dt
      COMPLEX vprop,tprop
      PARAMETER(npts=9,nptx=2**npts,NN=2)
      DIMENSION vprop(nptx,NN,NN),tprop(nptx)
c
      CALL ReadParam(nstep,ndump,dt)
      call Initialize()
      CALL SetKinProp(dt,tprop)
      CALL SetPotProp(dt,vprop)
      DO istep=1,nstep+1
         IF(mod(istep-1,10).EQ.0)
     1        PRINT *, "Step=", istep-1,", Final step=", nstep
         IF(istep.GE.1) CALL PROPAGATE(vprop,tprop)
         IF(mod((istep-1),ndump).EQ.0) THEN
            CALL SAVEWF(istep,ndump,dt)
         END IF
      END DO
 22   FORMAT(6(e13.6,2x))
      END
ccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccc
      SUBROUTINE energies(energy)
      IMPLICIT NONE
      INTEGER j,NN
      COMPLEX energy,RV,RKE
      PARAMETER (NN=2)
      DIMENSION RV(NN),RKE(NN),energy(NN)
      CALL PE(RV)
      CALL KE(RKE)
      DO j=1,NN
         energy(j)=RV(j)+RKE(j)
      END DO
      RETURN
      END
ccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccc
      subroutine ReadParam(nstep,ndump,dt)
c
c     Parameters defining the grid (xmin, xmax), integration time step (dt),
c     mass (amassx), initial position (xk), initial momentum (pk),
c     number of propagation steps (nstep), and how often to save a pic (ndump)
c
      IMPLICIT NONE
      INTEGER ntype,nstep,nrpt,ireport,ndump,nlit
      REAL xmin,xmax,pk,amassx,xk,dt
      common /packet/  amassx,xk,pk
      common /xy/  xmin,xmax
c
      xmin=-6.0
      xmax=6.0
      dt=0.2
      amassx=1.0
      xk=-2.2
      pk=0.
      nstep=100
      ndump=1
```

```
c
      return
      end
ccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccc
      SUBROUTINE Initialize()

      IMPLICIT NONE
      INTEGER NN,nptx,npts,kk
      COMPLEX chi0,chi,EYE,CRV
      REAL omega,xk2,xmin,xmax,dx,pi,amassx,xk,pk,x,alpha,alpha2
      PARAMETER(npts=9,nptx=2**npts,NN=2)
      DIMENSION CRV(NN,NN)
      common /xy/  xmin,xmax
      common /packet/  amassx,xk,pk
      COMMON / wfunc/ chi(nptx,NN)
      COMMON / iwfunc/ chi0(nptx,NN)

      EYE=(0.0,1.0)
      pi= acos(-1.0)
      omega=1.
      dx=(xmax-xmin)/real(nptx)
c
c     Wave Packet Initialization: Gaussian centered at xk, with momentum pk
c
      alpha=amassx*omega
      do kk=1,nptx
         x=xmin+kk*dx
         chi(kk,1)=((alpha/pi)**0.25)/sqrt(2.)
     1        *exp(-alpha/2.*(x-xk)**2+EYE*pk*(x-xk))
         chi(kk,2)=chi(kk,1)
c
         chi0(kk,1)=chi(kk,1)
         chi0(kk,2)=chi(kk,2)
      end do
      RETURN
      END
ccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccc
      SUBROUTINE SAVEWF(je2,ndump,dt)
c
c     Dump Time Evolved Wave packet
c
      IMPLICIT NONE
      INTEGER je2,nptx,npts,kk,NN,ncount,ndump,jj
      COMPLEX chi,CRV,energy,psi,Psia
      character*9 B
      REAL V,x1,c1,c2,c1a,x,xmin,xmax,dx,EVALUES,dt
      PARAMETER(npts=9,nptx=2**npts,NN=2)
      DIMENSION CRV(NN,NN),EVALUES(NN)
      DIMENSION psi(NN,NN)
      common /xy/  xmin,xmax
      COMMON / wfunc/ chi(nptx,NN)
      COMMON /ENER/ energy(NN)
c
      IF(je2.EQ.1) CALL energies(energy)
      jj=je2/ndump
      write(B, '(A,i4.4)') 'arch.', jj
      OPEN(1,FILE=B)
      dx=(xmax-xmin)/real(nptx)
      ncount=(je2-1)/ndump
c
c     Save Wave-packet components
```

```
c
      do kk=1,nptx
         x=xmin+kk*dx
         c1=chi(kk,1)*conjg(chi(kk,1))
         c2=chi(kk,2)*conjg(chi(kk,2))
         write(1,33) x,sqrt(c1)+real(energy(1))
      end do
      write(1,33)

      do kk=1,nptx
         x=xmin+kk*dx
         c2=chi(kk,2)*conjg(chi(kk,2))
          write(1,33) x,sqrt(c2)+real(energy(2))
      end do
      write(1,33)

      do kk=1,nptx
         x=xmin+kk*dx
          write(1,33) x,real(energy(2))
      end do
      write(1,33)

      do kk=1,nptx
         x=xmin+kk*dx
          write(1,33) x,real(energy(1))
      end do
      write(1,33)
c
c     Save Adiabatic states
c
      do kk=1,nptx
         x=xmin+kk*dx
         CALL HAMIL(CRV,x)
         CALL SCHROC1(CRV,psi,EVALUES)
         write(1,33) x,EVALUES(1)
      end do
      write(1,33)
      do kk=1,nptx
         x=xmin+kk*dx
         CALL HAMIL(CRV,x)
         CALL SCHROC1(CRV,psi,EVALUES)
         write(1,33) x,EVALUES(2)
      end do
      CLOSE(1)
 33   format(6(e13.6,2x))
      RETURN
      END
ccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccc
      subroutine SetKinProp(dt,tprop)
c
c     Kinetic Energy part of the Trotter Expansion: exp(-i p^2 dt/(2 m))
c
      IMPLICIT NONE
      INTEGER nptx,kx,nx,npts,NN
      REAL xsc,xmin,xmax,propfacx,amassx,xk,pi,alenx,dt,pk
      COMPLEX tprop,eye
      parameter(npts=9,nptx=2**npts,NN=2)
      DIMENSION tprop(nptx)
      common /xy/  xmin,xmax
      common /packet/  amassx,xk,pk
c
```

```
      eye=(0.,1.)
      pi = acos(-1.0)
      alenx=xmax-xmin
      propfacx=-dt/2./amassx*(2.*pi)**2
      do kx=1,nptx
         if(kx.le.(nptx/2+1)) then
            nx=kx-1
         else
            nx=kx-1-nptx
         end if
         xsc=0.
         if(nx.ne.0) xsc=real(nx)/alenx
         tprop(kx)=exp(eye*(propfacx*xsc**2))
       end do
c
      return
      end
ccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccc
      subroutine SetPotProp(dt,vprop)
c
c     Potential Energy part of the Trotter Expansion: exp(-i V dt/2)
c
      IMPLICIT NONE
      INTEGER NN,ii,kk,jj,nptx,i,j,k,npts
      REAL xmin,xmax,dx,dt,EVALUES,x
      COMPLEX vp,vprop,eye,dummy,psi,CRV
      parameter(npts=9,nptx=2**npts,NN=2)
      DIMENSION vprop(nptx,NN,NN),psi(NN,NN),CRV(NN,NN)
      DIMENSION vp(NN,NN),dummy(NN,NN),EVALUES(NN)
      common /xy/  xmin,xmax
      eye=(0.,1.)
      dx=(xmax-xmin)/real(nptx)
c
      do ii=1,nptx
         x=xmin+ii*dx
         CALL HAMIL(CRV,x)
         CALL SCHROC1(CRV,psi,EVALUES)
         vp(1,1)=exp(-eye*0.5*dt*EVALUES(1))
         vp(1,2)=0.0
         vp(2,1)=0.0
         vp(2,2)=exp(-eye*0.5*dt*EVALUES(2))
         do i=1,2
            do j=1,2
               dummy(i,j)=0.
               do k=1,2
                  dummy(i,j)=dummy(i,j)+vp(i,k)*psi(j,k)
               end do
            end do
         end do
         do i=1,2
            do j=1,2
               vp(i,j)=0.
               do k=1,2
                  vp(i,j)=vp(i,j)+psi(i,k)*dummy(k,j)
               end do
            end do
         end do
         do i=1,2
            do j=1,2
               kk=ii
               vprop(kk,i,j)=vp(i,j)/sqrt(1.0*nptx)
```

```
            end do
         end do
      end do
c
      RETURN
      END
ccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccc
      SUBROUTINE PROPAGATE(vprop,tprop)
c
c     Split Operator Fourier Transform Propagation Method
c     J. Comput. Phys. 47, 412 (1982); J. Chem. Phys. 78, 301 (1983)
c
      IMPLICIT NONE
      INTEGER i,j,kk,NN,in,ii,nptx,npts
      COMPLEX chi,vprop,chin1,chin2,tprop
      PARAMETER(npts=9,nptx=2**npts,NN=2)
      DIMENSION chin1(nptx),chin2(nptx)
      DIMENSION tprop(nptx),vprop(nptx,NN,NN)
      COMMON / wfunc/ chi(nptx,NN)
c
c     Apply potential energy part of the Trotter Expansion
c
      DO ii=1,nptx
         in=ii
         chin1(in)=0.0
         chin2(in)=0.0
         DO j=1,NN
            kk=ii
            chin1(in)=chin1(in)+vprop(kk,1,j)*chi(kk,j)
            chin2(in)=chin2(in)+vprop(kk,2,j)*chi(kk,j)
         END DO
      END DO
c
c     Fourier Transform wave-packet to the momentum representation
c
      CALL fourn(chin1,nptx,1,1)
      CALL fourn(chin2,nptx,1,1)
c
c     Apply kinetic energy part of the Trotter Expansion
c
      DO ii=1,nptx
         in=ii
         kk=ii
         chin1(in)=tprop(kk)*chin1(in)
         chin2(in)=tprop(kk)*chin2(in)
      END DO
c
c     Inverse Fourier Transform wave-packet to the coordinate representation
c
      CALL fourn(chin1,nptx,1,-1)
      CALL fourn(chin2,nptx,1,-1)
c
c     Apply potential energy part of the Trotter Expansion
c
      DO ii=1,nptx
         in=ii
         DO i=1,NN
            kk=ii
            chi(kk,i)=vprop(kk,i,1)*chin1(in)
     1           +vprop(kk,i,2)*chin2(in)
         END DO
```

```
      END DO
      END
ccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccc
      SUBROUTINE HAMIL(CRV,x)
c
c     Hamiltonian Matrix
c
      IMPLICIT NONE
      INTEGER NN
      REAL x,VPOT1,VPOT2
      COMPLEX CRV
      PARAMETER(NN=2)
      DIMENSION CRV(NN,NN)
c
      CALL VA(VPOT1,x)
      CALL VB(VPOT2,x)
      CRV(1,1)=VPOT1
      CRV(2,2)=VPOT2
      CRV(1,2)=0.3    ! comment this line for item (b)
      CRV(2,1)=0.3    ! comment this line for item (b)
c      CRV(1,2)=0.3   ! uncomment this line for item (b)
c      CRV(2,1)=0.3   ! uncomment this line for item (b)
c
      RETURN
      END
ccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccc
      SUBROUTINE VA(V,x)
c
c     Potential Energy Surface: Harmonic Oscillator
c
      implicit none
      REAL V,x,amassx,xk,pk,rk,omega
      common /packet/ amassx,xk,pk
      omega=1.0
      rk=amassx*omega**2
      V=0.5*rk*x*x
      RETURN
      END
ccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccc
      SUBROUTINE VB(V,x1)
c
c     Potential Energy Surface: Double-Well Potential, tunneling dynamics
c
      implicit none
      REAL V,x1,x
      x=x1
      V=-0.5*x**2+1.0/(16.0*1.3544)*x**4
      RETURN
      END

ccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccc
      SUBROUTINE PE(RV)
c
c     Expectation Value of the Potential Enegy
c
      IMPLICIT NONE
      INTEGER nptx,npts,kk,NN,j
      COMPLEX chi,EYE,RV
      REAL Vpot,omega,xmin,xmax,dx,pi,amassx,xk,pk,x,alpha
      PARAMETER(npts=9,nptx=2**npts,NN=2)
      DIMENSION RV(NN)
```

```
      COMMON / wfunc/ chi(nptx,NN)
      common /xy/  xmin,xmax
      common /packet/amassx,xk,pk

      dx=(xmax-xmin)/real(nptx)
      DO j=1,NN
         RV(j)=0.0
         do kk=1,nptx
            x=xmin+kk*dx
            IF(j.EQ.1) CALL VA(Vpot,x)
            IF(j.EQ.2) CALL VB(Vpot,x)
            RV(j)=RV(j)+chi(kk,j)*Vpot*conjg(chi(kk,j))*dx
         end do
      END DO
      RETURN
      END
ccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccc
      subroutine KE(RKE)
c
c     Expectation value of the kinetic energy
c
      IMPLICIT NONE
      INTEGER NN,kk,nptx,kx,nx,npts,j
      REAL dp,theta,wm,p,xmin,xmax,amassx,xk,pi,alenx,pk,rm,re,ri,dx
      COMPLEX eye,chi,Psip,chic,RKE
      parameter(npts=9,nptx=2**npts,NN=2)
      DIMENSION chic(nptx),RKE(NN)
      common /xy/  xmin,xmax
      common /packet/  amassx,xk,pk
      COMMON / wfunc/ chi(nptx,2)
c
      pi = acos(-1.0)
      dx=(xmax-xmin)/nptx
      dp=2.*pi/(xmax-xmin)
c
      DO j=1,NN
         RKE(j)=0.0
         do kk=1,nptx
            chic(kk)=chi(kk,j)
         end do
         CALL fourn(chic,nptx,1,1)
         do kx=1,nptx
            if(kx.le.(nptx/2+1)) then
               nx=kx-1
            else
               nx=kx-1-nptx
            end if
            p=0.
            if(nx.ne.0) p = real(nx)*dp
            chic(kx)=p**2/(2.0*amassx)*chic(kx)/nptx
         end do
         CALL fourn(chic,nptx,1,-1)
         do kk=1,nptx
            RKE(j)=RKE(j)+conjg(chi(kk,j))*chic(kk)*dx
         end do
      END DO
      return
      end
ccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccc
      SUBROUTINE SCHROC1(CRV,EVECT,EVALUES)
c
```

```
c     Hamiltonian Matrix Diagonalization
c
c     CRV: HERMITIAN MATRIX (INPUT)
c     EVECT: EIGENVECTORS (OUTPUT)
c     EVALUES: EIGENVALUES (OUTPUT)
c
      INTEGER N,I,J,NP
      REAL EVALUES,CRV2,EVECT2
      COMPLEX CRV,EVECT
      PARAMETER(N=2,NP=2)
      DIMENSION CRV(N,N),EVECT(N,N),EVALUES(N),E(NP)
      DIMENSION CRV2(N,N),EVECT2(N,N)
C
      DO I=1,N
         EVALUES(I)=0.0
         E(I)=0.0
         DO J=1,N
            CRV2(J,I)=CRV(J,I)
         END DO
      END DO
      CALL TRED2(CRV2,N,NP,EVALUES,E)
      CALL TQLI(EVALUES,E,N,NP,CRV2)
      CALL EIGSRT(EVALUES,CRV2,N,NP)
c
      DO I=1,N
         DO J=1,N
            EVECT(J,I)=CRV2(J,I)
         END DO
      END DO
C
      RETURN
      END
ccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccc
c     Subroutines from Numerical Recipes to compute FFT
ccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccc
      SUBROUTINE FOURN(DATA,NN,NDIM,ISIGN)
      REAL*8 WR,WI,WPR,WPI,WTEMP,THETA
      DIMENSION NN(NDIM),DATA(*)
      NTOT=1
      DO 11 IDIM=1,NDIM
         NTOT=NTOT*NN(IDIM)
 11   CONTINUE
      NPREV=1
      DO 18 IDIM=1,NDIM
         N=NN(IDIM)
         NREM=NTOT/(N*NPREV)
         IP1=2*NPREV
         IP2=IP1*N
         IP3=IP2*NREM
         I2REV=1
         DO 14 I2=1,IP2,IP1
            IF(I2.LT.I2REV)THEN
               DO 13 I1=I2,I2+IP1-2,2
                  DO 12 I3=I1,IP3,IP2
                     I3REV=I2REV+I3-I2
                     TEMPR=DATA(I3)
                     TEMPI=DATA(I3+1)
                     DATA(I3)=DATA(I3REV)
                     DATA(I3+1)=DATA(I3REV+1)
                     DATA(I3REV)=TEMPR
                     DATA(I3REV+1)=TEMPI
```

```
 12               CONTINUE
 13            CONTINUE
               ENDIF
               IBIT=IP2/2
 1             IF ((IBIT.GE.IP1).AND.(I2REV.GT.IBIT)) THEN
                  I2REV=I2REV-IBIT
                  IBIT=IBIT/2
                  GO TO 1
               ENDIF
               I2REV=I2REV+IBIT
 14         CONTINUE
            IFP1=IP1
 2          IF(IFP1.LT.IP2)THEN
               IFP2=2*IFP1
               THETA=ISIGN*6.28318530717959D0/(IFP2/IP1)
               WPR=-2.D0*DSIN(0.5D0*THETA)**2
               WPI=DSIN(THETA)
               WR=1.D0
               WI=0.D0
               DO 17 I3=1,IFP1,IP1
                  DO 16 I1=I3,I3+IP1-2,2
                     DO 15 I2=I1,IP3,IFP2
                        K1=I2
                        K2=K1+IFP1
                        TEMPR=SNGL(WR)*DATA(K2)-SNGL(WI)*DATA(K2+1)
                        TEMPI=SNGL(WR)*DATA(K2+1)+SNGL(WI)*DATA(K2)
                        DATA(K2)=DATA(K1)-TEMPR
                        DATA(K2+1)=DATA(K1+1)-TEMPI
                        DATA(K1)=DATA(K1)+TEMPR
                        DATA(K1+1)=DATA(K1+1)+TEMPI
 15                 CONTINUE
 16              CONTINUE
                 WTEMP=WR
                 WR=WR*WPR-WI*WPI+WR
                 WI=WI*WPR+WTEMP*WPI+WI
 17           CONTINUE
              IFP1=IFP2
              GO TO 2
           ENDIF
           NPREV=N*NPREV
 18    CONTINUE
       RETURN
       END
cccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccc
c     Subroutines to compute eigenvalues and eigenvectors
cccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccc
       SUBROUTINE TRED2(A,N,NP,D,E)
       IMPLICIT NONE
       INTEGER I,J,K,L,N,NP
       REAL A,D,E,H,SCALE,F,G,HH
       DIMENSION A(NP,NP),D(NP),E(NP)
       IF(N.GT.1)THEN
          DO 18 I=N,2,-1
             L=I-1
             H=0.
             SCALE=0.
             IF(L.GT.1)THEN
                DO 11 K=1,L
                   SCALE=SCALE+ABS(A(I,K))
 11             CONTINUE
                IF(SCALE.EQ.0.)THEN
```

```
                E(I)=A(I,L)
             ELSE
                DO 12 K=1,L
                   A(I,K)=A(I,K)/SCALE
                   H=H+A(I,K)**2
12              CONTINUE
                F=A(I,L)
                G=-SIGN(SQRT(H),F)
                E(I)=SCALE*G
                H=H-F*G
                A(I,L)=F-G
                F=0.
                DO 15 J=1,L
                   A(J,I)=A(I,J)/H
                   G=0.
                   DO 13 K=1,J
                      G=G+A(J,K)*A(I,K)
13                 CONTINUE
                   IF(L.GT.J)THEN
                      DO 14 K=J+1,L
                         G=G+A(K,J)*A(I,K)
14                    CONTINUE
                   ENDIF
                   E(J)=G/H
                   F=F+E(J)*A(I,J)
15              CONTINUE
                HH=F/(H+H)
                DO 17 J=1,L
                   F=A(I,J)
                   G=E(J)-HH*F
                   E(J)=G
                   DO 16 K=1,J
                      A(J,K)=A(J,K)-F*E(K)-G*A(I,K)
16                 CONTINUE
17              CONTINUE
             ENDIF
          ELSE
             E(I)=A(I,L)
          ENDIF
          D(I)=H
18     CONTINUE
     ENDIF
     D(1)=0.
     E(1)=0.
     DO 23 I=1,N
        L=I-1
        IF(D(I).NE.0.)THEN
           DO 21 J=1,L
              G=0.
              DO 19 K=1,L
                 G=G+A(I,K)*A(K,J)
19            CONTINUE
              DO 20 K=1,L
                 A(K,J)=A(K,J)-G*A(K,I)
20            CONTINUE
21         CONTINUE
        ENDIF
        D(I)=A(I,I)
        A(I,I)=1.
        IF(L.GE.1)THEN
           DO 22 J=1,L
```

```
              A(I,J)=0.
              A(J,I)=0.
 22         CONTINUE
         ENDIF
 23   CONTINUE
      RETURN
      END
ccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccc
      SUBROUTINE TQLI(D,E,N,NP,Z)
      IMPLICIT NONE
      INTEGER N,NP,I,K,L,M,ITER
      REAL D,E,Z,DD,G,R,S,C,P,F,B
      DIMENSION D(NP),E(NP),Z(NP,NP)
      IF (N.GT.1) THEN
         DO 11 I=2,N
            E(I-1)=E(I)
 11      CONTINUE
         E(N)=0.
         DO 15 L=1,N
            ITER=0
 1          DO 12 M=L,N-1
               DD=ABS(D(M))+ABS(D(M+1))
               IF (ABS(E(M))+DD.EQ.DD) GO TO 2
 12         CONTINUE
            M=N
 2          IF(M.NE.L)THEN
               IF(ITER.EQ.30) PAUSE 'too many iterations!'
               ITER=ITER+1
               G=(D(L+1)-D(L))/(2.*E(L))
               R=SQRT(G**2+1.)
               G=D(M)-D(L)+E(L)/(G+SIGN(R,G))
               S=1.
               C=1.
               P=0.
               DO 14 I=M-1,L,-1
                  F=S*E(I)
                  B=C*E(I)
                  IF(ABS(F).GE.ABS(G))THEN
                     C=G/F
                     R=SQRT(C**2+1.)
                     E(I+1)=F*R
                     S=1./R
                     C=C*S
                  ELSE
                     S=F/G
                     R=SQRT(S**2+1.)
                     E(I+1)=G*R
                     C=1./R
                     S=S*C
                  ENDIF
                  G=D(I+1)-P
                  R=(D(I)-G)*S+2.*C*B
                  P=S*R
                  D(I+1)=G+P
                  G=C*R-B
                  DO 13 K=1,N
                     F=Z(K,I+1)
                     Z(K,I+1)=S*Z(K,I)+C*F
                     Z(K,I)=C*Z(K,I)-S*F
 13               CONTINUE
 14            CONTINUE
```

```
              D(L)=D(L)-P
              E(L)=G
              E(M)=0.
              GO TO 1
            ENDIF
 15       CONTINUE
      ENDIF
      RETURN
      END
cccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccc
      SUBROUTINE EIGSRT(D,V,N,NP)
      IMPLICIT NONE
      INTEGER N,NP,I,J,K
      REAL D,V,P
      DIMENSION D(NP),V(NP,NP)
      DO 13 I=1,N-1
        K=I
        P=D(I)
        DO 11 J=I+1,N
          IF(D(J).GE.P)THEN
            K=J
            P=D(J)
          ENDIF
11      CONTINUE
        IF(K.NE.I)THEN
          D(K)=D(I)
          D(I)=P
          DO 12 J=1,N
            P=V(J,I)
            V(J,I)=V(J,K)
            V(J,K)=P
12        CONTINUE
        ENDIF
13    CONTINUE
      RETURN
      END
ccccccccccccccccccccccccccccccccccccccccccccccccccc
      SUBROUTINE PIKSRT(N,ARR)
      IMPLICIT NONE
      INTEGER I,J,N
      REAL ARR,A
      DIMENSION ARR(N)
      DO 12 J=2,N
        A=ARR(J)
        DO 11 I=J-1,1,-1
          IF(ARR(I).LE.A)GO TO 10
          ARR(I+1)=ARR(I)
11      CONTINUE
        I=0
10      ARR(I+1)=A
12    CONTINUE
      RETURN
      END
ccccccccccccccccccccccccccccccccccccccccccccccccccc
```