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

Download from (http://xbeams.chem.yale.edu/∼batista/P1/Problem1.f),

```
      PROGRAM Problem_1
      call Initialize()
      CALL SAVEWF(0)
      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,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
ccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccc
      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
```

ccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccc

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
```

Download from (http://xbeams.chem.yale.edu/∼batista/P2/Problem2.f),

```
      PROGRAM Problem2
      call Initialize()
      CALL SAVEFT()
      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,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
ccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccc
      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
cccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccc
```

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 $\langle\Psi_t|\hat{x}|\Psi_t\rangle$.

Download from (http://xbeams.chem.yale.edu/∼batista/P3/Problem3.f),

```
      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
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,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
ccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccc
      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
ccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccc
      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.

Download from (http://xbeams.chem.yale.edu/∼batista/P4/Problem4.f),

```
      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
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,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
ccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccc
      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
ccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccc
      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
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
c     Subroutines 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
ccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccc
```

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
```

Download from (http://xbeams.chem.yale.edu/∼batista/P6/Problem6.f),

```
      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
ccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccc
      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
ccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccc
      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
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=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
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,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
ccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccc
      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
ccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccc
      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
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
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
ccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccc
```

Problem 7:

In order to visualize the output of this program, cut the source code attached below, 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:

Download from (http://xbeams.chem.yale.edu/∼batista/P7/pp_7)

```
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
```

Download from (http://xbeams.chem.yale.edu/∼batista/P7/Problem7.f)

```
      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
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,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
ccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccc
      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
ccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccc
      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
ccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccc
      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
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 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 below, 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:

Download from (http://xbeams.chem.yale.edu/∼batista/P8/pp_8)

```
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
```

Download from (http://xbeams.chem.yale.edu/∼batista/P8/Problem8.f)

```
      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
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=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
ccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccc
      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
ccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccc
      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
ccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccc
      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
ccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccc
      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
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=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
ccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccc
      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
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=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
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=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
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
cccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccc
```

Problem 9:

The output of this program can be generated and visualized as follows. Cut the source code attached below, 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:

Download from (http://xbeams.chem.yale.edu/∼batista/P9/pp_9)

```
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
```

Download from (http://xbeams.chem.yale.edu/∼batista/P9/Problem9.f)

```
      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
ccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccc
      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
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=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
ccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccc
      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
ccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccc
      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
ccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccc
      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
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,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
ccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccc
      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
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=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
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=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
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=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 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:

According to the definition of the eigenstates of the potential energy matrix, given by Eq. (34),

$$\begin{pmatrix} V_{11} & V_{12} \\ V_{21} & V_{22} \end{pmatrix} \begin{pmatrix} L_{11} \\ L_{21} \end{pmatrix} = \begin{pmatrix} E_1 & 0 \\ 0 & E_1 \end{pmatrix} \begin{pmatrix} L_{11} \\ L_{21} \end{pmatrix}, \tag{14}$$

and

$$\begin{pmatrix} V_{11} & V_{12} \\ V_{21} & V_{22} \end{pmatrix} \begin{pmatrix} L_{12} \\ L_{22} \end{pmatrix} = \begin{pmatrix} E_2 & 0 \\ 0 & E_2 \end{pmatrix} \begin{pmatrix} L_{12} \\ L_{22} \end{pmatrix}. \tag{15}$$

Therefore,

$$\begin{pmatrix} V_{11} & V_{12} \\ V_{21} & V_{22} \end{pmatrix} \begin{pmatrix} L_{11} & L_{12} \\ L_{21} & L_{22} \end{pmatrix} = \begin{pmatrix} L_{11} & L_{12} \\ L_{21} & L_{22} \end{pmatrix} \begin{pmatrix} E_1 & 0 \\ 0 & E_2 \end{pmatrix}, \tag{16}$$

and

$$\begin{pmatrix} L_{11} & L_{21} \\ L_{12} & L_{22} \end{pmatrix} \begin{pmatrix} V_{11} & V_{12} \\ V_{21} & V_{22} \end{pmatrix} \begin{pmatrix} L_{11} & L_{12} \\ L_{21} & L_{22} \end{pmatrix} = \begin{pmatrix} E_1 & 0 \\ 0 & E_2 \end{pmatrix}, \tag{17}$$

or

$$\begin{pmatrix} V_{11} & V_{12} \\ V_{21} & V_{22} \end{pmatrix} = \begin{pmatrix} L_{11} & L_{12} \\ L_{21} & L_{22} \end{pmatrix} \begin{pmatrix} E_1 & 0 \\ 0 & E_2 \end{pmatrix}, \begin{pmatrix} L_{11} & L_{21} \\ L_{12} & L_{22} \end{pmatrix}. \tag{18}$$

Therefore, defining

$$\mathbf{L} = \begin{pmatrix} L_{11} & L_{21} \\ L_{12} & L_{22} \end{pmatrix}, \tag{19}$$

gives

$$\begin{pmatrix} V_{11} & V_{12} \\ V_{21} & V_{22} \end{pmatrix} = \mathbf{L}^{-1} \begin{pmatrix} E_1 & 0 \\ 0 & E_2 \end{pmatrix} \mathbf{L}. \tag{20}$$

Exponentiating both sides of Eq. (20), gives

$$e^{-i\tau \begin{pmatrix} V_{11} & V_{12} \\ V_{21} & V_{22} \end{pmatrix}} = e^{-i\tau \mathbf{L}^{-1} \begin{pmatrix} E_1 & 0 \\ 0 & E_2 \end{pmatrix} \mathbf{L}}. \tag{21}$$

Expanding the r.h.s. of Eq. (21) gives,

$$e^{-i\tau \begin{pmatrix} V_{11} & V_{12} \\ V_{21} & V_{22} \end{pmatrix}} = \begin{pmatrix} 1 & 0 \\ 0 & 1 \end{pmatrix} + \mathbf{L}^{-1} \begin{pmatrix} -i\tau E_1 & 0 \\ 0 & -i\tau E_2 \end{pmatrix} \mathbf{L} + \mathbf{L}^{-1} \begin{pmatrix} \frac{1}{2!} E_1^2 (-i\tau)^2 & 0 \\ 0 & \frac{1}{2!} E_2^2 (-i\tau)^2 \end{pmatrix} \mathbf{L} + ..., \tag{22}$$

since $\mathbf{L}^{-1}\mathbf{L} = 1$. Therefore,

$$e^{-i\tau \begin{pmatrix} V_{11} & V_{12} \\ V_{21} & V_{22} \end{pmatrix}} = \mathbf{L}^{-1} \begin{pmatrix} e^{-iE_1(\mathbf{x})\tau} & 0 \\ 0 & e^{-iE_2(\mathbf{x})\tau} \end{pmatrix} \mathbf{L}. \tag{23}$$

Problem 12:

The output of this program can be generated and visualized as follows. Cut the source code attached below, 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:

Download from (http://xbeams.chem.yale.edu/∼batista/P12/P12_c/pp_12)

```
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
```

Download from (http://xbeams.chem.yale.edu/∼batista/P12/P12_c/Problem12.f)

```
      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
ccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccc
      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
ccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccc
      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
ccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccc
      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
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=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
ccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccc
      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
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,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
ccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccc
      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
ccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccc
       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
ccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccc
       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
ccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccc
       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

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,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
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,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
```

```
ccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccc
      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
ccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccc
c     Subroutines to compute eigenvalues and eigenvectors
ccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccc
      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
cccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccc
      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
CCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCC
       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
CCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCC
       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
CCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCC
```

Problem 13:

The output of this program can be generated and visualized as follows. Download in the same directory the source code attached below from

http://xbeams.chem.yale.edu/∼batista/P13/P13.tar

and the math libraries from

http://xbeams.chem.yale.edu/∼batista/m.tar.

Untar both files by typing

```
tar -xvf P13.tar
```

and

```
tar -xvf m.tar
```

Type

```
cd P13
```

Compile the program with the script by typing

```
  comp_13
```

and run it by typing

```
  Problem13
```

.

Visualize the output as follows: type

```
gnuplot
```

then type

```
plot ``arch.0001''
```

That will show the matching representation of the amplitude of the target state, with one term in the expansion, and

```
replot ``arch.0001 u 1:3''
```

visualizes the real part of target state, also with one term in the expansion. The analytic results can be visualized on top by typing

```
replot ``arch.0001 u 1:4''
```

and

```
replot ``arch.0001 u 1:5''
```

, respectively. Note that since the potential is harmonic, the expansion with a single term is already converged. The results with two and three terms in the expansion can be visualized analogously by using arch.0002 and arch.0003, respectivley. To exit, type

```
quit
```

```
      Program Problem13
c
c     Generate a matching pursuit expansion of the target state
c     |\tilde{Psi}_0> = exp(-i p^2/(2m) tau/2) exp(-i V tau)
c                    exp(-i p^2/(2m) tau/2) |Psi_0>
c     where |Psi_0> is a Gaussian
c
      IMPLICIT NONE
      character*9 B
      INTEGER i,in,j,ISF,ID,npoints,maxbasis,NC,nta,NPT,ntraj,ndic
      REAL*8 dtv,dtt,dtp,mm,norm,normt,x,dx,xmin,xmax,x0,pi
      complex*16 xnc,pnc,FI,rnum,cg,gaussian,eye,cpc,x1,p1,g1
      complex*16 rt,it,rana,cdic,xdic,pdic,gdic
      PARAMETER(NC=1,NPT=4,nta=100,npoints=100)
      DIMENSION x(nc),normt(2),rnum(npoints),mm(NC),pdic(nta,nc)
      DIMENSION x1(nc),p1(nc),g1(nc),cdic(nta),xdic(nta,nc),gdic(nta,nc)
      common /NUCLEAR/ xnc(nta,NC,NPT),pnc(nta,NC,NPT)
      common /NUC/ cpc(nta,NPT),FI(nta,NC,NPT),ntraj(NPT)
c
      eye=(0.0d0, 1.0d0)
      pi=dacos(-1.0d0)
      mm(1)=1.0
c
c     Initialize the wavepacket as a single Gaussian
c
      do i=1,NPT
         ntraj(i)=0
      enddo
      ntraj(1)=1    ! Number of terms in the expansion of the initial state
      cpc(1,1)=1.0  ! Expansion coefficients
c
      DO in=1,NC
         xnc(1,in,1) = -2.5  ! Position of initial state
         pnc(1,in,1) = 0.0   ! Momentum of initial state
         FI(1,in,1)  = 1.0    ! Width of initial state
      ENDDO
c
c     Propagation time increments for Trotter expansion
c
      dtt = 0.1
      dtv = dtt
      dtp = dtt/2.0d0
c
c     Initialize dictionary for the Matching Pursuit.
c
      isf=1
      ID=isf*2-1
      ndic=ntraj(ID)       ! number of basis functions at t(ID)
      do i=1,ndic
         do in=1,NC
            xdic(i,in) = xnc(i,in,ID)
            pdic(i,in) = pnc(i,in,ID)
            gdic(i,in) = FI(i,in,ID)
         enddo
      enddo
c
c     Output Initial State
c
      do in=1,NC
         x1(in) = xnc(1,in,ID)
         p1(in) = pnc(1,in,ID)
```

```
         g1(in) = FI(1,in,ID)
         print *, "Dimension",in
         print *, "xpg 0",x1(in),p1(in),g1(in)
       enddo
c
       cg=cpc(1,ID)              ! expansion coefficient at initial time t(ID)
       print *, "coef 0",cg
c
c      Save initial wavepacket in file step0
c
       open(unit=100,file="step0")
       xmin  =-10.0
       xmax  = 10.0
       dx=(xmax-xmin)/(npoints-1)
       do i=1,npoints
          x(1)=xmin+dx*(i-1)
          write (100,222) x(1),dreal(cg*gaussian(x,x1,p1,g1))
      $        ,imag(cg*gaussian(x,x1,p1,g1))
       enddo
       close(100)
c
c      Obtain target state |\tilde{Psi}_0> = exp(-i p^2/(2m) tau/2)
c                exp(-i V tau) exp(-i p^2/(2m) tau/2) |Psi_0>
c      as a MP coherent-state expansion
c
       ntraj(ID+1)=0        ! number of basis functions at t(ID+1)
       maxbasis=3           ! maximum # of basis functions in the dictionary
c
       call Match_Pursuit(norm,isf,ndic,gdic,xdic,pdic,
      $     cdic,maxbasis,dtv,dtp,mm)
c
c      Output MP wavepacket after finding each term by sequential
c      orthogonal decomposition
c
       x0=-2.5
       xmin=-10.0
       xmax=10.0
       dx=(xmax-xmin)/(npoints-1)
c
       do i=1,npoints
          rnum(i)=0.
       end do
c
       DO j=1,maxbasis
          write(B, '(A,i4.4)') 'arch.', j
          open(100,file=B)
          do in=1,NC
             x1(in) = xnc(j,in,ID+1)
             p1(in) = pnc(j,in,ID+1)
             g1(in) = FI(j,in,ID+1)
          enddo
          cg=cpc(j,ID+1)
c
c      Save MP wavepacket in file step1
c
          do i=1,npoints
             x(1)=xmin+dx*(i-1)
c
c      Analytic wavepacket for comparision
c
             rt=-(x(1)-x0*cos(0.1))**2/2.0
```

```
            it=sin(0.1)*(x0**2*cos(0.1)-2.0*x(1)*x0)/2.0
            rana=(1.0/pi)**0.25*(cos(-0.05)+eye*sin(-0.05))*
     $           cdexp(rt+eye*it)
            rnum(i)=rnum(i)+cg*gaussian(x,x1,p1,g1)
            write (100,222) x(1),dreal(rnum(i)),dimag(rnum(i))
     $           ,dreal(rana),dimag(rana)
         enddo
         close(1)
       END DO
 222   format(8(e13.6,2x))
       end
ccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccc
       SUBROUTINE Match_Pursuit(norm,isf,ndic,gdic,xdic,pdic,
     $     cdic,maxbasis,dtv,dtp,mm)
c
       IMPLICIT NONE
       INTEGER i,in,k,j,ISF,ID,maxbasis,ntraj,ndic
       INTEGER imp,Nmax,NC,nta,NPT
       REAL*8 dtv,dtvc,dtp,mm,rcut,c1,c2,norm
       complex*16 x1,p1,x2,p2,g1,g2,xdic,pdic
       complex*16 gdic,cdic,xnc,pnc,FI,EYE,cpc,ovl,precoef,gij
       PARAMETER(rcut=1.E-8,NC=1,NPT=4,nta=100)
       DIMENSION x1(NC),p1(NC),g1(NC),x2(NC),p2(NC),g2(NC)
       DIMENSION mm(nc),xdic(nta,NC),pdic(nta,NC),gdic(nta,NC),cdic(nta)
       common /NUCLEAR/ xnc(nta,NC,NPT),pnc(nta,NC,NPT)
       common /NUC/ cpc(nta,NPT),FI(nta,NC,NPT),ntraj(NPT)
c
       EYE=(0.0,1.0)
       id=isf*2                  ! index of optimized c-state
c
c      calculate the overlap cdic for each item in the dictionary.
c
       call overlap(ndic,gdic,xdic,pdic,cdic,ISF,dtv,dtp,mm)
       imp=0
       norm=0.0
 10    continue
c
C      Find the item of the dictionary that is the best match
c
       Nmax=1    ! Nmax is the index of the best match
       do i=1,ndic
          c1=abs(cdic(i))
          c2=abs(cdic(Nmax))
          if (c1.gt.c2) Nmax=i
       enddo
c
C      Use the best match as the initial guess for further optimization
c
       precoef=cdic(Nmax)
       imp=imp+1                      ! index of the term in the expansion
       do in=1,NC
          xnc(imp,in,ID)=xdic(Nmax,in)
          pnc(imp,in,ID)=pdic(Nmax,in)
          FI(imp,in,ID)=gdic(Nmax,in)
       enddo
       cpc(imp,ID)=precoef
c
       call optimize(imp,isf,dtv,dtp,mm)
       ntraj(ID)=imp
       c1=conjg(cpc(imp,ID))*cpc(imp,ID)
       norm=norm+c1
```

```
c
c     Cutoff criteria
c
c     if (c1.lt.rcut) goto 27
      if (imp.ge.maxbasis) goto 27 ! maxbasis tells the cutout for expansion
c
c     Compute expansion coefficients
c
      do in=1,NC
         x2(in)=xnc(imp,in,ID)
         p2(in)=pnc(imp,in,ID)
         g2(in)=FI(imp,in,ID)
      enddo

      do i=1,ndic
         do in=1,NC
            x1(in)=xdic(i,in)
            p1(in)=pdic(i,in)
            g1(in)=gdic(i,in)
         enddo
         call overlap_ggovlc(x1,p1,g1,x2,p2,g2,gij)        !  gij=<1|2>
         cdic(i)=cdic(i)-cpc(imp,ID)*gij         !   expansion coefficient
      enddo
      goto 10
 27   return
      end
C-------------------------------------------------------------------------
      subroutine optimize(imp,ISF,dtv,dtp,mm)
c
C     Gradient-based optimization subroutine to maximize the overlap between
c     the imp-th target function and the trial coherent state
c     which is returned in the common blocks
C     common /NUCLEAR/ xnc(nta,NC,NPT),pnc(nta,NC,NPT)
c     common /NUC/ cpc(nta,NPT),FI(nta,NC,NPT),ntraj(NPT)
c
      implicit none
      integer i,j,in,Nmax,imp,Ndiv,Ntrial
      integer iter,ntraj,diter,ditermax,giter,gitermax
      integer NPROC,me,ierr,rc
      integer ISF,ID
      integer NC,nta,NPT
      PARAMETER(NC=1,NPT=4,nta=100)
      real*8 dx,rr,al,al0,ali,ala,alb,alc,ald,dtv,dtvc,dtp
      real*8 rtr,rtar,gain,ratio,almax,expect,norm,up,down,mm(nc)
      complex*16 xp,pp,gp
      complex*16 xa,pa,ga,qa,xb,pb,gb,qb,xc,pc,gc,qc,qd
      complex*16 dr,dp,dg
      complex*16 r1,p1,g1,r2,p2,g2
      complex*16 rm(nta,NC),pm(nta,NC),gm(nta,NC),qm(nta)
      complex*16 xnc,pnc,FI
      complex*16 qsum,c2,c1,c0,c3,c4
      complex*16 gij,ovl,qmp,cpc,qp,eye
      dimension r1(NC),p1(NC),g1(NC),r2(NC),p2(NC),g2(NC)
      dimension dr(NC),dp(NC),dg(NC),rr(6*NC)
      dimension xp(NC),pp(NC),gp(NC)
      dimension xa(NC),pa(NC),ga(NC),xb(NC),pb(NC),gb(NC)
      dimension xc(NC),pc(NC),gc(NC)
      common /NUCLEAR/ xnc(nta,NC,NPT),pnc(nta,NC,NPT)
      common /NUC/ cpc(nta,NPT),FI(nta,NC,NPT),ntraj(NPT)
c
      ID=2*ISF
```

```
      ntrial=6
      eye=(0.0,1.0)
      do in=1,NC
         r1(in)=xnc(imp,in,ID)
         p1(in)=pnc(imp,in,ID)
         g1(in)=FI(imp,in,ID)
      enddo
      c0=cpc(imp,ID)
      c1=c0
      almax=0.0
      ditermax=0
      gitermax=0
      iter=0
 9    iter=iter+1
      do in=1,NC
         xa(in)=r1(in)
         pa(in)=p1(in)
         ga(in)=g1(in)
      enddo
      qa=c1
      ala=0.0
c
c     Computes the partial derivatives of the overlap with respect to
c     the adjustable CS parameters
c
      call Derivative(r1,p1,g1,c1,rr,ISF,dtv,dtp,mm)
c
      do in=1,NC
         dr(in)=rr(0*NC+in)+rr(1*NC+in)*eye
         dp(in)=rr(2*NC+in)+rr(3*NC+in)*eye
         dg(in)=rr(4*NC+in)+rr(5*NC+in)*eye
         dg(in)=0.0
      enddo
      rtr=0.0
      do in=1,6*NC
         rtr=rtr+rr(in)*rr(in)
      enddo
      rtr=sqrt(rtr)
      if (rtr.eq.0.0) goto 10
      al=abs(c1)/rtr
      if (al.gt.8.0) al=8.0
      if (al.lt.1.0e-1) al=1.0e-1
      al0=al
      diter=0
 15   diter=diter+1
      if ((diter-1)*Ntrial.gt.24) goto 10
c
c     Incrementing parameters along the direction of the gradients
c
 16   do i=1,Ntrial
         ali=al/2.0**(Ntrial-i)
         do in=1,NC
            rm(i,in)=r1(in)+dr(in)/rtr*ali
            pm(i,in)=p1(in)+dp(in)/rtr*ali
            gm(i,in)=g1(in)+dg(in)/rtr*ali
            if (dreal(gm(i,in)).lt.0.0) then
               al=al/2.0
               al0=al
               goto 16
            endif
            if (dreal(gm(i,in)).lt.dreal(g1(in))*0.3) then
```

```
               al=al/2.0
               al0=al
               goto 16
            endif
         enddo
      enddo
c
      call overlap(ntrial,gm,rm,pm,qm,ISF,dtv,dtp,mm)
c
c     Select the maximum
c
      if (al.gt.almax) almax=al
      Nmax=1
      do i=1,Ntrial
         if (abs(qm(i)).gt.abs(qm(Nmax))) Nmax=i
      enddo
      c2=qm(Nmax)
      if (abs(c2).le.abs(c1)) then
         al=al/2.0**Ntrial
         goto 15
      endif
      if (diter.gt.ditermax) ditermax=diter
      alb=al/2.0**(Ntrial-Nmax)
      qb=c2
      if (giter.gt.gitermax) gitermax=giter
      ratio=(abs(qb)-abs(c1))/abs(c1)
      if (ratio.gt.0.0) then
         do in=1,NC
            r1(in)=r1(in)+dr(in)/rtr*alb
            p1(in)=p1(in)+dp(in)/rtr*alb
            g1(in)=g1(in)+dg(in)/rtr*alb
         enddo
         c1=qb
      endif
      if ((abs(qb)-abs(c1)).gt.1.0E-5) goto 9
      if (ratio.lt.0.001) goto 10
      if (iter.gt.NC*2) goto 10
      goto 9
 10   continue
c
c     Update trial parameters with optimized parameters
c
      if (abs(c1).gt.abs(c0)) then
         do in=1,NC
            xnc(imp,in,ID)=r1(in)
            pnc(imp,in,ID)=p1(in)
            FI(imp,in,ID)=g1(in)
         enddo
         cpc(imp,ID)=c1
      endif
      qp=cpc(imp,ID)
      gain=(abs(qp)/abs(c0))**2
      return
      end
C-----------------------------------------------------------------------
      subroutine Derivative(rin,pin,gin,c0,rr,ISF,dtv,dtp,mm)
c
c     Computes the partial derivatives of the overlap with respect to
c     the adjustable CS parameters
c
      implicit none
```

```
      integer i,k,in,Ndiv,ISF,ID
      integer NPROC,me,ierr,ntraj,nt
      integer NC,nta,NPT
      PARAMETER(NC=1,NPT=4,nta=100)
      real*8 dx,rr,dtv,dtvc,dtp,mm(nc)
      complex*16 x1,p1,g1,x2,p2,g2,rin,pin,gin
      complex*16 c0,c1,eye,gvgovl,gvgovl_id,gvgovly2
      complex*16 xnc,pnc,cpc,FI,ggovl,ggovl_id,ggovlc
      complex*16 rm(nta,NC),pm(nta,NC),gm(nta,NC),qm(nta)
      COMMON /NUCLEAR/ xnc(nta,NC,NPT),pnc(nta,NC,NPT)
      common /NUC/ cpc(nta,NPT),FI(nta,NC,NPT),ntraj(NPT)
      dimension rin(NC),pin(NC),gin(NC)
      dimension x1(NC),p1(NC),g1(NC),x2(NC),p2(NC),g2(NC)
      dimension rr(6*NC)
      dimension gvgovl_id(nta*nta),gvgovl(nta*nta)
      dimension ggovl_id(nta*nta),ggovl(nta*nta)
c
      eye=(0.0,1.0)
      dx=0.001
      do in=1,6*NC
         rr(in)=0.0
      enddo
      do i=1,6*NC
         do in=1,NC
            rm(i,in)=rin(in)
            pm(i,in)=pin(in)
            gm(i,in)=gin(in)
         enddo
         qm(i)=0.0
      enddo
      do in=1,NC
         k=0*NC+in
         rm(k,in)=rm(k,in)+dx
         k=1*NC+in
         rm(k,in)=rm(k,in)+eye*dx
         k=2*NC+in
         pm(k,in)=pm(k,in)+dx
         k=3*NC+in
         pm(k,in)=pm(k,in)+eye*dx
         k=4*NC+in
         gm(k,in)=gm(k,in)+dx
         k=5*NC+in
         gm(k,in)=gm(k,in)+eye*dx
      enddo
      nt=6*NC
      call overlap(nt,gm,rm,pm,qm,ISF,dtv,dtp,mm)
      do i=1,6*NC
         rr(i)=(abs(qm(i))-abs(c0))/dx
      enddo
      return
      end
C-------------------------------------------------------------------------------
      subroutine overlap(ndic,gdic,xdic,pdic,cdic,ISF,dtv,dtp,mm)
c
C     Find out which cs from the dictionary has maximum
c     overlap with the target function
c
      IMPLICIT NONE
      integer NPROC,me,ierr,ndiv,nta,NPT,ntraj,I,in,NC
      integer ISF,index_dic,index_ntraj,id,ndic,nmp,idx
      integer index_ntraj12,isfc,idc
```

```
      PARAMETER(NC=1,NPT=4,nta=100)
      real*8 dtv,dtvc,dtp,mm(nc)
      complex*16 g1(nc),g2(nc),x1(nc)
      complex*16 x2(nc),p1(nc),p2(nc)
      complex*16 xnc,pnc,cpc,FI,gvgovlc,ggovlc,cdic1(nta)
      complex*16 gdic(nta,nc),xdic(nta,nc),pdic(nta,nc),cdic(nta)
      common /NUCLEAR/ xnc(nta,NC,NPT),pnc(nta,NC,NPT)
      common /NUC/ cpc(nta,NPT),FI(nta,NC,NPT),ntraj(NPT)
c
      id=2*isf-1
      do i=1,ndic
         cdic(i)=0.0D0
         cdic1(i)=0.0D0
      enddo
      do index_ntraj=1,ntraj(id)
         do index_dic=1,ndic
            do in=1,NC                          ! trial coherent-state
               g1(in)=gdic(index_dic,in)
               p1(in)=pdic(index_dic,in)
               x1(in)=xdic(index_dic,in)
            enddo
            do in=1,NC          ! expansion terms in the ket_{index_ntraj
               x2(in)=xnc(index_ntraj,in,ID)
               p2(in)=pnc(index_ntraj,in,ID)
               g2(in)= FI(index_ntraj,in,ID)
            enddo
c
c     <trial|Trotter exp.|ket_{index_ntraj}>
c
            call overlap_gexpvg_g1_coupling
     $           (x1,p1,g1,x2,p2,g2,gvgovlc,dtv,dtp,mm)
c
c     <trial|Trotter exp.|target>
c
            cdic1(index_dic)=cdic1(index_dic)+
     $           cpc(index_ntraj,ID)*gvgovlc
            if ((ntraj(ID+1).ge.1.).AND.(index_ntraj.EQ.1)) then
               do i=1,ntraj(ID+1)
                  do in=1,NC
                     x2(in)=xnc(i,in,ID+1)
                     p2(in)=pnc(i,in,ID+1)
                     g2(in)=FI(i,in,ID+1)
                  enddo
                  call overlap_ggovlc(x1,p1,g1,x2,p2,g2,ggovlc)
c
c     <trial|Trotter exp.|residue>
c
                  cdic1(index_dic)=cdic1(index_dic)-cpc(i,ID+1)*ggovlc
               enddo
            endif
         enddo
      enddo
      do i=1,ndic
         cdic(i)=cdic1(i)
      enddo
      return
      end
C-------------------------------------------------------------------------
      subroutine overlap_gexpvg_g1_coupling
     $     (x1,p1,g1,x2,p2,g2,gvgovl,dtv,dtp,mm)
c
```

```
c     Calculatea <CS_1| exp(-i K dt/2)*exp(-i V dt)*exp(-i K dt/2)|CS_2 >
c
      IMPLICIT NONE
      INTEGER NG,nx,ny,IND,J,NFLAG,I,in,JJ,Ngd,ISF
      integer NC,nta,NPT,ngrid,isfc
      PARAMETER(NC=1,NPT=4,nta=100)
      REAL*8 mm(nc),dtvc,dtp,pi,dtv
      real*8 x(nc),z(nc),VPOT,xi,wi,xg,wgd
      real*8 a,b,c,d,e,f,dtp1
      real*8 a1,b1,c1,d1,e1,f1
      real*8 a2,b2,c2,d2,e2,f2
      complex*16 x1,x2,p1,p2,g1,g2,gf1,gf2,gaussian_type2,expvc
      complex*16 aa,bb,cc,den,aa1,bb1,cc1,aa2,bb2,cc2,N1,N2
      COMPLEX*16 ovl,ovl1,GF,eye,gvgovl,fx,yovl,gaussian
      real*8  xpro(NC),xmax(NC),xmin(NC),dx(NC)
      dimension x1(NC),x2(NC),p1(NC),p2(NC),g1(NC),g2(NC)
      dimension a(NC),b(NC),c(NC),d(NC),e(NC),f(NC)
      dimension aa(NC),bb(NC),cc(NC)
      dimension a1(NC),b1(NC),c1(NC),d1(NC),e1(NC),f1(NC)
      dimension a2(NC),b2(NC),c2(NC),d2(NC),e2(NC),f2(NC)

      integer jn
      real*8 conv

      complex*16 coefAs1(nc,nc),coefBs1(nc),coefCs1
      complex*16 coefAs2(nc,nc),coefBs2(nc),coefCs2
      complex*16 coefAc(nc,nc),coefBc(nc),coefCc

      complex*16 coefA1(nc,nc),coefB1(nc),coefC1
      complex*16 coefA2(nc,nc),coefB2(nc),coefC2

      complex*16 caa1(nc,nc),cbb1(nc),ccc1
      complex*16 caa2(nc,nc),cbb2(nc),ccc2

      integer dim,incx,incy,info,IPIV(nc),ifail
      character*1 trans
      complex*16 zdotu,y(nc),ia(nc,nc),F06GAF
      complex*16 overlap1,overlap2,wkspacei(nc),alpha,beta
      real*8 detr,deti,wkspace(nc)
      integer IPIVOT(nc),job
      complex work(nc),det(2),sia(nc,nc)
c
      dtp1=-dtp
      pi=dacos(-1.0d0)
      eye=(0.0,1.0)
c
      coefCs1=0.0
      coefBs1(1)=0.0
      coefAs1(1,1)=0.5
c
      coefC1=-eye*dtv*coefCs1
      do i=1,nc
         coefB1(i)=-eye*dtv*coefBs1(i)
         do j=1,nc
            coefA1(i,j)=-eye*dtv*coefAs1(i,j)
         enddo
      enddo
c
      ccc1=coefC1
      N1=1.0
      N2=1.0
```

```
c
      do in=1,NC
         a1(in)=dreal(g1(in))
         b1(in)=dimag(g1(in))
         c1(in)=dreal(x1(in))
         d1(in)=dimag(x1(in))
         e1(in)=dreal(p1(in))
         f1(in)=dimag(p1(in))

         a2(in)=dreal(g2(in))
         b2(in)=dimag(g2(in))
         c2(in)=dreal(x2(in))
         d2(in)=dimag(x2(in))
         e2(in)=dreal(p2(in))
         f2(in)=dimag(p2(in))
c
c     Normalization constants
c
         N1=N1*(a1(in)/pi)**0.25*exp(-0.5*a1(in)*d1(in)**2
     &        -d1(in)*e1(in)-(b1(in)*d1(in)+f1(in))**2/2.0/a1(in))
     &        *sqrt(mm(in)/(mm(in)+eye*dtp1*g1(in)))
         N2=N2*(a2(in)/pi)**0.25*exp(-0.5*a2(in)*d2(in)**2
     &        -d2(in)*e2(in)-(b2(in)*d2(in)+f2(in))**2/2.0/a2(in))
     &        *sqrt(mm(in)/(mm(in)+eye*dtp*g2(in)))
c
c     Integrand=N2*exp(aa2*x^2+cc2*x+cc2)*conjg(N1*exp(aa1*x^2+cc1*x+cc1))
c                   *exp(ccc1+cbb1*x+caa1*x^2)
c
         den=2.0+2.0*eye*dtp*g2(in)/mm(in)
         aa2=-g2(in)/den
         bb2=(2.0*eye*p2(in)+2.0*g2(in)*x2(in))/den
         cc2=(p2(in)-eye*g2(in)*x2(in))**2/g2(in)/den
     &        -p2(in)**2/2.0/g2(in)

         den=2.0+2.0*eye*dtp1*g1(in)/mm(in)
         aa1=-g1(in)/den
         bb1=(2.0*eye*p1(in)+2.0*g1(in)*x1(in))/den
         cc1=(p1(in)-eye*g1(in)*x1(in))**2/g1(in)/den
     &        -p1(in)**2/2.0/g1(in)

         ccc1=ccc1+dconjg(cc1)+cc2
         cbb1(in)=dconjg(bb1)+bb2+coefB1(in)
         do jn=1,nc
            if (in.eq.jn) then
               caa1(in,jn)=dconjg(aa1)+aa2+coefA1(in,jn)
            else
               caa1(in,jn)=coefA1(in,jn)
            endif
         enddo
      enddo
      dim=nc
      do i=1,nc
         y(i)=0.0
         cbb1(i)=-cbb1(i)
         do j=1,nc
            caa1(i,j)=-caa1(i,j)
            ia(i,j)=caa1(i,j)
            sia(i,j)=ia(i,j)
         enddo
      enddo
c
```

```
c     NAG subroutines
c
c     call F03ADF(caa1,dim,dim,detr,deti,wkspace,ifail)
c     overlap1=dsqrt(pi**dim)/cdsqrt(detr+eye*deti)
      job=11
c
c     SGI subroutines for computations of the determinant
c
      call CGEFA(sIA,dim,dim,IPIVOT,INFO)
      CALL CGEdi(sIA, dim, dim, IPIVOT, DET, WORK, JOB)
      overlap1=dsqrt(pi**dim)/sqrt(det(1)*10.0**det(2))
c
c     call F07ARF(dim,dim,IA,dim,IPIV,info)
      call zgetrf(dim,dim,IA,dim,IPIV,info)
c     call F07AWF(dim,IA,dim,IPIV,wkspacei,dim,info)
      call zgetri(dim,IA,dim,IPIV,wkspacei,dim,info)
c
      trans='N'
      alpha=1.0d0
      beta=0.0d0
      do i=1,dim
         y(i)=0.0d0
      enddo
      incx=1     !  Matrix multiplication for exponent of the G-integral
      incy=1
c     call F06SAF(trans,dim,dim,alpha,IA,dim,cbb1,incx,beta,y,incy)
c     overlap1=overlap1*cdexp(F06GAF(dim,cbb1,incx,y,incy)/4.0d0+ccc1)
      call zgemv(trans,dim,dim,alpha,IA,dim,cbb1,incx,beta,y,incy)
      overlap1=dconjg(N1)*N2
     $     *overlap1*cdexp(zdotu(dim,cbb1,incx,y,incy)/4.0d0+ccc1)
      gvgovl=overlap1
      RETURN
      END
C-------------------------------------------------------------------------------
      subroutine overlap_ggovlc(r1,p1,g1,r2,p2,g2,govl)
c
c     calculate <r1,p1,g1|r2,p2,g2> analytically, the parameters
c     are complex numbers
c
      implicit none
      integer in,NC,nta,NPT
      PARAMETER(NC=1,NPT=4,nta=100)
      real*8 pi,a1,b1,c1,d1,e1,f1,a2,b2,c2,d2,e2,f2
      complex*16 r1,r2,p1,p2,g1,g2
      complex*16 N1,N2,aa1,bb1,cc1,aa2,bb2,cc2,aa,bb,cc
      complex*16 phi22,eye,govl
      dimension r1(NC),r2(NC),p1(NC),p2(NC),g1(NC),g2(NC)
      dimension a1(NC),b1(NC),c1(NC),d1(NC),e1(NC),f1(NC)
      dimension a2(NC),b2(NC),c2(NC),d2(NC),e2(NC),f2(NC)
c
      eye=(0.0,1.0)
      pi=3.141592654
      N1=1.0
      N2=1.0
      phi22=1.0
      do in=1,NC
         a1(in)=dreal(g1(in))
         b1(in)=dimag(g1(in))
         c1(in)=dreal(r1(in))
         d1(in)=dimag(r1(in))
         e1(in)=dreal(p1(in))
```

```
         f1(in)=dimag(p1(in))
         a2(in)=dreal(g2(in))
         b2(in)=dimag(g2(in))
         c2(in)=dreal(r2(in))
         d2(in)=dimag(r2(in))
         e2(in)=dreal(p2(in))
         f2(in)=dimag(p2(in))
         N1=N1*(a1(in)/pi)**0.25*exp(-0.5*a1(in)*d1(in)**2
     &        -d1(in)*e1(in)-(b1(in)*d1(in)+f1(in))**2/2.0/a1(in))
         N2=N2*(a2(in)/pi)**0.25*exp(-0.5*a2(in)*d2(in)**2
     &        -d2(in)*e2(in)-(b2(in)*d2(in)+f2(in))**2/2.0/a2(in))
         aa1=-0.5*g1(in)
         bb1=g1(in)*r1(in)+eye*p1(in)
         cc1=-0.5*g1(in)*r1(in)**2-eye*p1(in)*r1(in)
         aa2=-0.5*g2(in)
         bb2=g2(in)*r2(in)+eye*p2(in)
         cc2=-0.5*g2(in)*r2(in)**2-eye*p2(in)*r2(in)
         aa=conjg(aa1)+aa2
         bb=conjg(bb1)+bb2
         cc=conjg(cc1)+cc2
         phi22=phi22*exp(-bb**2/4.0/aa+cc)
         phi22=phi22*sqrt(-pi/aa)
         if (dreal(aa).gt.0.0) then
            print *,"r1=",r1(in)
            print *,"r1=",p1(in)
            print *,"r1=",g1(in)
            print *,"r2=",r2(in)
            print *,"p2=",p2(in)
            print *,"g2=",g2(in)
            print *,"aa=",aa
            print *,"error"
            stop
         endif
       enddo
       phi22=phi22*conjg(N1)*N2
       if (abs(phi22).gt.1.0E20) phi22=0.0
       if (abs(phi22).lt.1.0E-20) phi22=0.0
       govl=phi22
       return
       end
C-------------------------------------------------------------------------------
       FUNCTION gaussian(x,x1,p1,g1)
c
c      Gaussian basis fucntion
c
       IMPLICIT NONE
       INTEGER in
       integer NC,nta,NPT
       PARAMETER(NC=1,NPT=4,nta=100)

       REAL*8 x
       real*8 pi,a1,b1,c1,d1,e1,f1
       complex*16 x1,p1,g1
       COMPLEX*16 EYE,GAU,gaussian,N1
       DIMENSION x(NC),x1(NC),p1(NC),g1(NC)
       dimension a1(NC),b1(NC),c1(NC),d1(NC),e1(NC),f1(NC)
c
       pi=3.141592654
       EYE=(0.0,1.0)
c
       do in=1,NC
```

```
         a1(in)=dreal(g1(in))
         b1(in)=dimag(g1(in))
         c1(in)=dreal(x1(in))
         d1(in)=dimag(x1(in))
         e1(in)=dreal(p1(in))
         f1(in)=dimag(p1(in))
       enddo
c
       N1=1.0
       do in=1,NC
          N1=N1*(a1(in)/pi)**0.25*exp(-0.5*a1(in)*d1(in)**2
     &         -d1(in)*e1(in)-(b1(in)*d1(in)+f1(in))**2/2.0/a1(in))
       enddo
c
       GAU=1.0
       DO in=1,NC
          GAU=GAU*EXP(-0.5*g1(in)*(x(in)-x1(in))**2
     &         +EYE*p1(in)*(x(in)-x1(in)))
       END DO
       GAU=GAU*N1
c
       if (abs(gau).gt.1.0E20) gau=0.0
       if (abs(gau).lt.1.0E-20) gau=0.0
       gaussian=gau
c
       RETURN
       END
C-------------------------------------------------------------------------------
       FUNCTION gaussian_type2(x,x1,p1,g1,dt,m)
c
c      Gaussian basis function operated by the kinetic operator
c
       IMPLICIT NONE
       INTEGER in
       integer NC,nta,NPT
       PARAMETER(NC=1,NPT=4,nta=100)
       REAL*8 x,pi,m,dt
       real*8 a1,b1,c1,d1,e1,f1
       complex*16 x1,p1,g1
       COMPLEX*16 EYE,GAU2,rnum,rden,gaussian_type2,N1
       DIMENSION x(NC),x1(NC),p1(NC),g1(NC),m(NC)
       dimension a1(NC),b1(NC),c1(NC),d1(NC),e1(NC),f1(NC)
c
       pi=3.141592654
       EYE=(0.0,1.0)
c
       do in=1,NC
          a1(in)=dreal(g1(in))
          b1(in)=dimag(g1(in))
          c1(in)=dreal(x1(in))
          d1(in)=dimag(x1(in))
          e1(in)=dreal(p1(in))
          f1(in)=dimag(p1(in))
       enddo
c
       N1=1.0
       do in=1,NC
          N1=N1*(a1(in)/pi)**0.25*exp(-0.5*a1(in)*d1(in)**2
     &         -d1(in)*e1(in)-(b1(in)*d1(in)+f1(in))**2/2.0/a1(in))
       enddo
c
```

```
      GAU2=1.0
      DO in=1,NC
         rnum=p1(in)/g1(in)-EYE*(x1(in)-x(in))
         rden=2.0/g1(in)+EYE*2.0*dt/m(in)
         GAU2=GAU2*EXP(rnum**2/rden-p1(in)**2/(2.0*g1(in)))
     &        *sqrt(m(in)/(m(in)+eye*g1(in)*dt))
      END DO
      GAU2=GAU2*N1
      if (abs(gau2).gt.1.0E20) gau2=0.0
      if (abs(gau2).lt.1.0E-20) gau2=0.0
      gaussian_type2=gau2
      RETURN
      END
c-----------------------------------------------------------------------
```

Problem 14:

The output of this program can be generated and visualized as follows. Download in the same directory the source code attached below from

http://xbeams.chem.yale.edu/∼batista/P14/P14.tar

and the math libraries from

http://xbeams.chem.yale.edu/∼batista/m.tar.

Untar both files by typing

```
tar -xvf P14.tar
```

and

```
tar -xvf m.tar
```

Type

```
cd P14
```

Compile the program with the script by typing

```
comp_14
```

and run it by typing

```
Problem14
```

.

The snapshots of the time-dependent wave-packet can be visualized by compiling the program plot.f by executing plot_14, running the plot executable and then displaying the movie by typing

```
gnuplot<pp_14
```

where the file named

```
pp_14
```

has the following lines:

```
set yrange[-1:1]
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 Problem14
c
c     MP/SOFT propagation
c
      IMPLICIT NONE
      character*9 B
      INTEGER i,in,j,ISF,ID,npoints,maxbasis,NC,nta,NPT,ntraj,ndic,nstep
      REAL*8 dtv,dtt,dtp,mm,norm,normt,x,dx,xmin,xmax,x0,pi
      complex*16 xnc,pnc,FI,rnum,cg,gaussian,eye,cpc,x1,p1,g1
      complex*16 rt,it,rana,cdic,xdic,pdic,gdic
      PARAMETER(NC=1,NPT=2,nta=100,npoints=100)
      DIMENSION x(nc),normt(2),rnum(npoints),mm(NC),pdic(nta,nc)
      DIMENSION x1(nc),p1(nc),g1(nc),cdic(nta),xdic(nta,nc),gdic(nta,nc)
      common /NUCLEAR/ xnc(nta,NC,NPT),pnc(nta,NC,NPT)
      common /NUC/ cpc(nta,NPT),FI(nta,NC,NPT),ntraj(NPT)
c
      character*30 num,name
c
      eye=(0.0d0, 1.0d0)
      pi=dacos(-1.0d0)
      mm(1)=1.0
c
c     Initialize the wavepacket as a single Gaussian
c
      do i=1,NPT
         ntraj(i)=0
      enddo
      ntraj(1)=1                ! Number of terms in the initial expansion
      cpc(1,1)=1.0              ! Expansion coefficients
c
      DO in=1,NC
         xnc(1,in,1) = -2.5     ! Position of initial state
         pnc(1,in,1) = 0.0      ! Momentum of initial state
         FI(1,in,1)  = 1.0      ! Width of initial state
      ENDDO
c
c     Propagation increments for the Trotter expansion
c
      dtt = 0.1
      dtv = dtt
      dtp = dtt/2.0d0
      nstep=20                  ! propagation step.
      maxbasis=10               ! maximum # of basis functions in the dict.
c
      j=0
      call number_to_char(j,num)
      name="reinit."//num
      open(2,file=name)
      write(2,24) j,ntraj(1),dtt,norm
      do i=1,ntraj(1)
         do in=1,NC
            write(2,22) xnc(i,in,1),pnc(i,in,1),FI(i,in,1)
         enddo
         write(2,22) cpc(i,1),abs(cpc(i,1))**2
      enddo
         write(2,22)
      close(2)
c
c     Propagation loop
c
      do j=1,nstep
```

```
         isf=1
         ID=isf*2-1
         ndic=ntraj(ID)         ! number of basis functions at t(ID)

         do i=1,ndic
            do in=1,NC
               xdic(i,in) = xnc(i,in,ID)
               pdic(i,in) = pnc(i,in,ID)
               gdic(i,in) = FI(i,in,ID)
            enddo
c     print *, "xpg",xdic(i,1),pdic(i,1),gdic(i,1)
         enddo
         print *, "step ", j, "in propagation"
         ntraj(ID+1)=0          ! number of basis functions at t(ID+1)
c
         call Match_Pursuit(norm,isf,ndic,gdic,xdic,pdic,
     $        cdic,maxbasis,dtv,dtp,mm)

         do i=1,ntraj(ID+1)     ! Update Wave Function
            do in=1,NC
               xnc(i,in,ID)=xnc(i,in,ID+1)
               pnc(i,in,ID)=pnc(i,in,ID+1)
               FI(i,in,ID)=FI(i,in,ID+1)
            enddo
            cpc(i,ID)=cpc(i,ID+1)
         enddo
         ntraj(ID)=ntraj(ID+1)

         call number_to_char(j,num)
         name="reinit."//num
         open(2,file=name)
         write(2,24) j,ntraj(1),dtt,norm
         do i=1,ntraj(1)
            do in=1,NC
               write(2,22) xnc(i,in,1),pnc(i,in,1),FI(i,in,1)
            enddo
            write(2,22) cpc(i,1),abs(cpc(i,1))**2
         enddo
         close(2)
      enddo
 22   FORMAT(6(e13.6,2x))
 24   format(I6,2x,I6,2x,e13.6,2x,e13.6)
      end
ccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccc
      SUBROUTINE Match_Pursuit(norm,isf,ndic,gdic,xdic,pdic,
     $     cdic,maxbasis,dtv,dtp,mm)
c
      IMPLICIT NONE
      INTEGER i,in,k,j,ISF,ID,maxbasis,ntraj,ndic
      INTEGER imp,Nmax,NC,nta,NPT
      REAL*8 dtv,dtvc,dtp,mm,rcut,c1,c2,norm
      complex*16 x1,p1,x2,p2,g1,g2,xdic,pdic
      complex*16 gdic,cdic,xnc,pnc,FI,EYE,cpc,ovl,precoef,gij
      PARAMETER(rcut=1.E-8,NC=1,NPT=2,nta=100)
      DIMENSION x1(NC),p1(NC),g1(NC),x2(NC),p2(NC),g2(NC)
      DIMENSION mm(nc),xdic(nta,NC),pdic(nta,NC),gdic(nta,NC),cdic(nta)
      common /NUCLEAR/ xnc(nta,NC,NPT),pnc(nta,NC,NPT)
      common /NUC/ cpc(nta,NPT),FI(nta,NC,NPT),ntraj(NPT)
c
      EYE=(0.0,1.0)
```

```
      id=isf*2
c
c     calculate the overlap cdic for each item in the dictionary.
c
      call overlap(ndic,gdic,xdic,pdic,cdic,ISF,dtv,dtp,mm)
      imp=0
      norm=0.0
 10   continue

c      print *, "isf",isf
c      print *, "ndic",ndic
c      print *, cdic(1)
c      print *,gdic(1,1),xdic(1,1),pdic(1,1)

c
C     Find the item of the dictionary that is the best match
c
      Nmax=1    ! Nmax is the index of the best match
      do i=1,ndic
         c1=abs(cdic(i))
         c2=abs(cdic(Nmax))
         if (c1.gt.c2) Nmax=i
      enddo
c
C     Use the best match as the initial guess for further optimization
c
      precoef=cdic(Nmax)
      imp=imp+1
      do in=1,NC
         xnc(imp,in,ID)=xdic(Nmax,in)
         pnc(imp,in,ID)=pdic(Nmax,in)
         FI(imp,in,ID)=gdic(Nmax,in)
      enddo
      cpc(imp,ID)=precoef
c
      call optimize(imp,isf,dtv,dtp,mm)
      ntraj(ID)=imp
      c1=conjg(cpc(imp,ID))*cpc(imp,ID)
      norm=norm+c1
c
c     Cutoff criteria
c
c     if (c1.lt.rcut) goto 27
      if (imp.ge.maxbasis) goto 27 ! maxbasis tells the cutout for expansion
c
c     Compute expansion coefficients
c
      do in=1,NC
         x2(in)=xnc(imp,in,ID)
         p2(in)=pnc(imp,in,ID)
         g2(in)=FI(imp,in,ID)
      enddo

      do i=1,ndic
         do in=1,NC
            x1(in)=xdic(i,in)
            p1(in)=pdic(i,in)
            g1(in)=gdic(i,in)
         enddo
         call overlap_ggovlc(x1,p1,g1,x2,p2,g2,gij)  ! gij=<1|2>
         cdic(i)=cdic(i)-cpc(imp,ID)*gij             ! expansion coefficient
```

```
      enddo
      goto 10
 27   return
      end
ccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccc
      subroutine optimize(imp,ISF,dtv,dtp,mm)
c
C     Gradient-based optimization subroutine to maximize the overlap between
c     the target function and the imp-th coherent state
c     which is returned in the common blocks
C     common /NUCLEAR/ xnc(nta,NC,NPT),pnc(nta,NC,NPT)
c     common /NUC/ cpc(nta,NPT),FI(nta,NC,NPT),ntraj(NPT)
c
      implicit none
      integer i,j,in,Nmax,imp,Ndiv,Ntrial
      integer iter,ntraj,diter,ditermax,giter,gitermax
      integer NPROC,me,ierr,rc
      integer ISF,ID
      integer NC,nta,NPT
      PARAMETER(NC=1,NPT=2,nta=100)
      real*8 dx,rr,al,al0,ali,ala,alb,alc,ald,dtv,dtvc,dtp
      real*8 rtr,rtar,gain,ratio,almax,expect,norm,up,down,mm(nc)
      complex*16 xp,pp,gp
      complex*16 xa,pa,ga,qa,xb,pb,gb,qb,xc,pc,gc,qc,qd
      complex*16 dr,dp,dg
      complex*16 r1,p1,g1,r2,p2,g2
      complex*16 rm(nta,NC),pm(nta,NC),gm(nta,NC),qm(nta)
      complex*16 xnc,pnc,FI
      complex*16 qsum,c2,c1,c0,c3,c4
      complex*16 gij,ovl,qmp,cpc,qp,eye
      dimension r1(NC),p1(NC),g1(NC),r2(NC),p2(NC),g2(NC)
      dimension dr(NC),dp(NC),dg(NC),rr(6*NC)
      dimension xp(NC),pp(NC),gp(NC)
      dimension xa(NC),pa(NC),ga(NC),xb(NC),pb(NC),gb(NC)
      dimension xc(NC),pc(NC),gc(NC)
      common /NUCLEAR/ xnc(nta,NC,NPT),pnc(nta,NC,NPT)
      common /NUC/ cpc(nta,NPT),FI(nta,NC,NPT),ntraj(NPT)
      ID=2*ISF
      ntrial=6
      eye=(0.0,1.0)
      do in=1,NC
         r1(in)=xnc(imp,in,ID)
         p1(in)=pnc(imp,in,ID)
         g1(in)=FI(imp,in,ID)
      enddo
      c0=cpc(imp,ID)
      c1=c0
      almax=0.0
      ditermax=0
      gitermax=0
      iter=0
 9    iter=iter+1
      do in=1,NC
         xa(in)=r1(in)
         pa(in)=p1(in)
         ga(in)=g1(in)
      enddo
      qa=c1
      ala=0.0
      call Derivative(r1,p1,g1,c1,rr,ISF,dtv,dtp,mm)
      do in=1,NC
```

```
         dr(in)=rr(0*NC+in)+rr(1*NC+in)*eye
         dp(in)=rr(2*NC+in)+rr(3*NC+in)*eye
         dg(in)=rr(4*NC+in)+rr(5*NC+in)*eye
         dg(in)=0.0
      enddo
      rtr=0.0
      do in=1,6*NC
         rtr=rtr+rr(in)*rr(in)
      enddo
      rtr=sqrt(rtr)
      if (rtr.eq.0.0) goto 10
      al=abs(c1)/rtr
      if (al.gt.8.0) al=8.0
      if (al.lt.1.0e-1) al=1.0e-1
      al0=al
      diter=0
15    diter=diter+1
      if ((diter-1)*Ntrial.gt.24) goto 10
16    do i=1,Ntrial
         ali=al/2.0**(Ntrial-i)
         do in=1,NC
            rm(i,in)=r1(in)+dr(in)/rtr*ali
            pm(i,in)=p1(in)+dp(in)/rtr*ali
            gm(i,in)=g1(in)+dg(in)/rtr*ali
            if (dreal(gm(i,in)).lt.0.0) then
               al=al/2.0
               al0=al
               goto 16
            endif
            if (dreal(gm(i,in)).lt.dreal(g1(in))*0.3) then
               al=al/2.0
               al0=al
               goto 16
            endif
         enddo
      enddo
      call overlap
     $     (ntrial,gm,rm,pm,qm,ISF,dtv,dtp,mm)

      if (al.gt.almax) almax=al
      Nmax=1
      do i=1,Ntrial
         if (abs(qm(i)).gt.abs(qm(Nmax))) Nmax=i
      enddo
      c2=qm(Nmax)
      if (abs(c2).le.abs(c1)) then
         al=al/2.0**Ntrial
         goto 15
      endif
      if (diter.gt.ditermax) ditermax=diter

      alb=al/2.0**(Ntrial-Nmax)
      qb=c2
      if (giter.gt.gitermax) gitermax=giter
      ratio=(abs(qb)-abs(c1))/abs(c1)
      if (ratio.gt.0.0) then
         do in=1,NC
            r1(in)=r1(in)+dr(in)/rtr*alb
            p1(in)=p1(in)+dp(in)/rtr*alb
            g1(in)=g1(in)+dg(in)/rtr*alb
         enddo
```

```
         c1=qb
      endif
      if ((abs(qb)-abs(c1)).gt.1.0E-5) goto 9
      if (ratio.lt.0.001) goto 10
      if (iter.gt.NC*2) goto 10
      goto 9
 10   continue
      if (abs(c1).gt.abs(c0)) then
         do in=1,NC
            xnc(imp,in,ID)=r1(in)
            pnc(imp,in,ID)=p1(in)
            FI(imp,in,ID)=g1(in)
         enddo
         cpc(imp,ID)=c1
      endif
      qp=cpc(imp,ID)
      gain=(abs(qp)/abs(c0))**2
 22   format(8(e13.6,2x))
      return
      end
ccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccc
      subroutine Derivative(rin,pin,gin,c0,rr,ISF,dtv,dtp,mm)
c
c     Computes the partial derivatives of the overlap with respect to
c     the adjustable CC parameters
c
      implicit none
      integer i,k,in,Ndiv,ISF,ID
      integer NPROC,me,ierr,ntraj,nt
      integer NC,nta,NPT
      PARAMETER(NC=1,NPT=2,nta=100)
      real*8 dx,rr,dtv,dtvc,dtp,mm(nc)
      complex*16 x1,p1,g1,x2,p2,g2,rin,pin,gin
      complex*16 c0,c1,eye,gvgovl,gvgovl_id,gvgovly2
      complex*16 xnc,pnc,cpc,FI,ggovl,ggovl_id,ggovlc
      complex*16 rm(nta,NC),pm(nta,NC),gm(nta,NC),qm(nta)
      COMMON /NUCLEAR/ xnc(nta,NC,NPT),pnc(nta,NC,NPT)
      common /NUC/ cpc(nta,NPT),FI(nta,NC,NPT),ntraj(NPT)
      dimension rin(NC),pin(NC),gin(NC)
      dimension x1(NC),p1(NC),g1(NC),x2(NC),p2(NC),g2(NC)
      dimension rr(6*NC)
      dimension gvgovl_id(nta*nta),gvgovl(nta*nta)
      dimension ggovl_id(nta*nta),ggovl(nta*nta)
c
      eye=(0.0,1.0)
      dx=0.001
      do in=1,6*NC
         rr(in)=0.0
      enddo
      do i=1,6*NC
         do in=1,NC
            rm(i,in)=rin(in)
            pm(i,in)=pin(in)
            gm(i,in)=gin(in)
         enddo
         qm(i)=0.0
      enddo
      do in=1,NC
         k=0*NC+in
         rm(k,in)=rm(k,in)+dx
         k=1*NC+in
```

```
         rm(k,in)=rm(k,in)+eye*dx
         k=2*NC+in
         pm(k,in)=pm(k,in)+dx
         k=3*NC+in
         pm(k,in)=pm(k,in)+eye*dx
         k=4*NC+in
         gm(k,in)=gm(k,in)+dx
         k=5*NC+in
         gm(k,in)=gm(k,in)+eye*dx
      enddo
      nt=6*NC
      call overlap
     $     (nt,gm,rm,pm,qm,ISF,dtv,dtp,mm)

      do i=1,6*NC
         rr(i)=(abs(qm(i))-abs(c0))/dx
      enddo
      return
      end
ccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccc
      subroutine overlap(ndic,gdic,xdic,pdic,cdic,ISF,dtv,dtp,mm)
c
C     Find out which cc from the dictionary has maximum
c     overlap with the target function
c
      IMPLICIT NONE
      integer NPROC,me,ierr,ndiv,nta,NPT,ntraj,I,in,NC
      integer ISF,index_dic,index_ntraj,id,ndic,nmp,idx
      integer index_ntraj12,isfc,idc
      PARAMETER(NC=1,NPT=2,nta=100)
      real*8 dtv,dtvc,dtp,mm(nc)
      complex*16 g1(nc),g2(nc),x1(nc)
      complex*16 x2(nc),p1(nc),p2(nc)
      complex*16 xnc,pnc,cpc,FI,gvgovlc,ggovlc,cdic1(nta)
      complex*16 gdic(nta,nc),xdic(nta,nc),pdic(nta,nc),cdic(nta)
      common /NUCLEAR/ xnc(nta,NC,NPT),pnc(nta,NC,NPT)
      common /NUC/ cpc(nta,NPT),FI(nta,NC,NPT),ntraj(NPT)
c
      id=2*isf-1
      do i=1,ndic
         cdic(i)=0.0D0
         cdic1(i)=0.0D0
      enddo
      do index_ntraj=1,ntraj(id)
         do index_dic=1,ndic
            do in=1,NC
               g1(in)=gdic(index_dic,in)
               p1(in)=pdic(index_dic,in)
               x1(in)=xdic(index_dic,in)
            enddo
            do in=1,NC
               x2(in)=xnc(index_ntraj,in,ID)
               p2(in)=pnc(index_ntraj,in,ID)
               g2(in)= FI(index_ntraj,in,ID)
            enddo
            call overlap_gexpvg_g1_coupling
     $           (x1,p1,g1,x2,p2,g2,gvgovlc,dtv,dtp,mm)

c            print *, "gv",gvgovlc
            cdic1(index_dic)=cdic1(index_dic)+
     $           cpc(index_ntraj,ID)*gvgovlc
```

```
            if ((ntraj(ID+1).ge.1.).AND.(index_ntraj.EQ.1)) then
               do i=1,ntraj(ID+1)
                  do in=1,NC
                     x2(in)=xnc(i,in,ID+1)
                     p2(in)=pnc(i,in,ID+1)
                     g2(in)=FI(i,in,ID+1)
                  enddo
                  call overlap_ggovlc(x1,p1,g1,x2,p2,g2,ggovlc)
                  cdic1(index_dic)=cdic1(index_dic)-cpc(i,ID+1)*ggovlc
               enddo
            endif
         enddo
      enddo
      do i=1,ndic
         cdic(i)=cdic1(i)
      enddo

c      print *,"cdic", cdic(1)
      return
      end
ccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccc
      subroutine number_to_char(filenumber,dotb)
      implicit none
      integer i,filenumber,number,jk,num(5)
      character dotb*5,c,name*10

      number=filenumber
      jk=0
      do i=1,5
         dotb(i:i)=" "
      enddo
      do while (number.gt.9)
         i=mod(number,10)
         jk=jk+1
         num(jk)=ichar('0')+i
         number=number/10
      enddo
      jk=jk+1
      num(jk)=ichar('0')+number

      do i=1,jk
         c=char(num(i))
         dotb(jk-i+1:jk-i+1)=c
      enddo
      return
      end
ccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccc
      subroutine overlap_gexpvg_g1_coupling
     $     (x1,p1,g1,x2,p2,g2,gvgovl,dtv,dtp,mm)
c
c     Calculatea <CS_1|exp(-iKdt/2)*exp(-iVdt)*exp(-iKdt/2)|CS_2>
c
      IMPLICIT NONE
      INTEGER NG,nx,ny,IND,J,NFLAG,I,in,JJ,Ngd,ISF
      integer NC,nta,NPT,ngrid,isfc
      PARAMETER(NC=1,NPT=4,nta=100)
      REAL*8 mm(nc),dtvc,dtp,pi,dtv
      real*8 x(nc),z(nc),VPOT,xi,wi,xg,wgd
      real*8 a,b,c,d,e,f,dtp1
      real*8 a1,b1,c1,d1,e1,f1
```

```
      real*8 a2,b2,c2,d2,e2,f2
      complex*16 x1,x2,p1,p2,g1,g2,gf1,gf2,gaussian_type2,expvc
      complex*16 aa,bb,cc,den,aa1,bb1,cc1,aa2,bb2,cc2,N1,N2
      COMPLEX*16 ovl,ovl1,GF,eye,gvgovl,fx,yovl,gaussian
      real*8  xpro(NC),xmax(NC),xmin(NC),dx(NC)
      dimension x1(NC),x2(NC),p1(NC),p2(NC),g1(NC),g2(NC)
      dimension a(NC),b(NC),c(NC),d(NC),e(NC),f(NC)
      dimension aa(NC),bb(NC),cc(NC)
      dimension a1(NC),b1(NC),c1(NC),d1(NC),e1(NC),f1(NC)
      dimension a2(NC),b2(NC),c2(NC),d2(NC),e2(NC),f2(NC)

      integer jn
      real*8 conv

      complex*16 coefAs1(nc,nc),coefBs1(nc),coefCs1
      complex*16 coefAs2(nc,nc),coefBs2(nc),coefCs2
      complex*16 coefAc(nc,nc),coefBc(nc),coefCc

      complex*16 coefA1(nc,nc),coefB1(nc),coefC1
      complex*16 coefA2(nc,nc),coefB2(nc),coefC2

      complex*16 caa1(nc,nc),cbb1(nc),ccc1
      complex*16 caa2(nc,nc),cbb2(nc),ccc2

      integer dim,incx,incy,info,IPIV(nc),ifail
      character*1 trans
      complex*16 zdotu,y(nc),ia(nc,nc),F06GAF
      complex*16 overlap1,overlap2,wkspacei(nc),alpha,beta
      real*8 detr,deti,wkspace(nc)
      integer IPIVOT(nc),job
      complex work(nc),det(2),sia(nc,nc)
c
      dtp1=-dtp
      pi=dacos(-1.0d0)
      eye=(0.0,1.0)
c
      coefCs1=0.0
      coefBs1(1)=0.0
      coefAs1(1,1)=0.5
c
      coefC1=-eye*dtv*coefCs1
      do i=1,nc
         coefB1(i)=-eye*dtv*coefBs1(i)
         do j=1,nc
            coefA1(i,j)=-eye*dtv*coefAs1(i,j)
         enddo
      enddo
c
      ccc1=coefC1
      N1=1.0
      N2=1.0
c
      do in=1,NC
         a1(in)=dreal(g1(in))
         b1(in)=dimag(g1(in))
         c1(in)=dreal(x1(in))
         d1(in)=dimag(x1(in))
         e1(in)=dreal(p1(in))
         f1(in)=dimag(p1(in))

         a2(in)=dreal(g2(in))
```

```
          b2(in)=dimag(g2(in))
          c2(in)=dreal(x2(in))
          d2(in)=dimag(x2(in))
          e2(in)=dreal(p2(in))
          f2(in)=dimag(p2(in))
c
c     Normalization constants
c
          N1=N1*(a1(in)/pi)**0.25*exp(-0.5*a1(in)*d1(in)**2
     &         -d1(in)*e1(in)-(b1(in)*d1(in)+f1(in))**2/2.0/a1(in))
     &         *sqrt(mm(in)/(mm(in)+eye*dtp1*g1(in)))
          N2=N2*(a2(in)/pi)**0.25*exp(-0.5*a2(in)*d2(in)**2
     &         -d2(in)*e2(in)-(b2(in)*d2(in)+f2(in))**2/2.0/a2(in))
     &         *sqrt(mm(in)/(mm(in)+eye*dtp*g2(in)))
c
c     Integrand=N2*exp(aa2*x^2+cc2*x+cc2)*conjg(N1*exp(aa1*x^2+cc1*x+cc1))
c              *exp(ccc1+cbb1*x+caa1*x^2)
c
          den=2.0+2.0*eye*dtp*g2(in)/mm(in)
          aa2=-g2(in)/den
          bb2=(2.0*eye*p2(in)+2.0*g2(in)*x2(in))/den
          cc2=(p2(in)-eye*g2(in)*x2(in))**2/g2(in)/den
     &         -p2(in)**2/2.0/g2(in)

          den=2.0+2.0*eye*dtp1*g1(in)/mm(in)
          aa1=-g1(in)/den
          bb1=(2.0*eye*p1(in)+2.0*g1(in)*x1(in))/den
          cc1=(p1(in)-eye*g1(in)*x1(in))**2/g1(in)/den
     &         -p1(in)**2/2.0/g1(in)

          ccc1=ccc1+dconjg(cc1)+cc2
          cbb1(in)=dconjg(bb1)+bb2+coefB1(in)
          do jn=1,nc
             if (in.eq.jn) then
                caa1(in,jn)=dconjg(aa1)+aa2+coefA1(in,jn)
             else
                caa1(in,jn)=coefA1(in,jn)
             endif
          enddo
       enddo
       dim=nc
       do i=1,nc
          y(i)=0.0
          cbb1(i)=-cbb1(i)
          do j=1,nc
             caa1(i,j)=-caa1(i,j)
             ia(i,j)=caa1(i,j)
             sia(i,j)=ia(i,j)
          enddo
       enddo
c
c     NAG subroutines
c
c     call F03ADF(caa1,dim,dim,detr,deti,wkspace,ifail)
c     overlap1=dsqrt(pi**dim)/cdsqrt(detr+eye*deti)
       job=11
c
c     SGI subroutines to compute the terminant
c
       call CGEFA(sIA,dim,dim,IPIVOT,INFO)
       CALL CGEdi(sIA, dim, dim, IPIVOT, DET, WORK, JOB)
```

```
      overlap1=dsqrt(pi**dim)/sqrt(det(1)*10.0**det(2))
c
c     call F07ARF(dim,dim,IA,dim,IPIV,info)
      call zgetrf(dim,dim,IA,dim,IPIV,info)
c     call F07AWF(dim,IA,dim,IPIV,wkspacei,dim,info)
      call zgetri(dim,IA,dim,IPIV,wkspacei,dim,info)
c
      trans='N'
      alpha=1.0d0
      beta=0.0d0
      do i=1,dim
         y(i)=0.0d0
      enddo
      incx=1     !  Matrix multiplication for exponent
      incy=1
c     call F06SAF(trans,dim,dim,alpha,IA,dim,cbb1,incx,beta,y,incy)
c     overlap1=overlap1*cdexp(F06GAF(dim,cbb1,incx,y,incy)/4.0d0+ccc1)
      call zgemv(trans,dim,dim,alpha,IA,dim,cbb1,incx,beta,y,incy)
      overlap1=dconjg(N1)*N2
     $     *overlap1*cdexp(zdotu(dim,cbb1,incx,y,incy)/4.0d0+ccc1)
      gvgovl=overlap1
      RETURN
      END
C-------------------------------------------------------------------------
      subroutine overlap_ggovlc(r1,p1,g1,r2,p2,g2,govl)
c
c     calculate <r1,p1,g1|r2,p2,g2> analytically, the parameters
c     are complex numbers
c
      implicit none
      integer in
      integer NC,nta,NPT
      PARAMETER(NC=1,NPT=4,nta=100)
      real*8 pi,a1,b1,c1,d1,e1,f1,a2,b2,c2,d2,e2,f2
      complex*16 r1,r2,p1,p2,g1,g2
      complex*16 N1,N2,aa1,bb1,cc1,aa2,bb2,cc2,aa,bb,cc
      complex*16 phi22,eye,govl
      dimension r1(NC),r2(NC),p1(NC),p2(NC),g1(NC),g2(NC)
      dimension a1(NC),b1(NC),c1(NC),d1(NC),e1(NC),f1(NC)
      dimension a2(NC),b2(NC),c2(NC),d2(NC),e2(NC),f2(NC)
c
      eye=(0.0,1.0)
      pi=3.141592654
      N1=1.0
      N2=1.0
      phi22=1.0
      do in=1,NC
         a1(in)=dreal(g1(in))
         b1(in)=dimag(g1(in))
         c1(in)=dreal(r1(in))
         d1(in)=dimag(r1(in))
         e1(in)=dreal(p1(in))
         f1(in)=dimag(p1(in))
         a2(in)=dreal(g2(in))
         b2(in)=dimag(g2(in))
         c2(in)=dreal(r2(in))
         d2(in)=dimag(r2(in))
         e2(in)=dreal(p2(in))
         f2(in)=dimag(p2(in))
         N1=N1*(a1(in)/pi)**0.25*exp(-0.5*a1(in)*d1(in)**2
     &        -d1(in)*e1(in)-(b1(in)*d1(in)+f1(in))**2/2.0/a1(in))
```

```
         N2=N2*(a2(in)/pi)**0.25*exp(-0.5*a2(in)*d2(in)**2
     &        -d2(in)*e2(in)-(b2(in)*d2(in)+f2(in))**2/2.0/a2(in))
         aa1=-0.5*g1(in)
         bb1=g1(in)*r1(in)+eye*p1(in)
         cc1=-0.5*g1(in)*r1(in)**2-eye*p1(in)*r1(in)
         aa2=-0.5*g2(in)
         bb2=g2(in)*r2(in)+eye*p2(in)
         cc2=-0.5*g2(in)*r2(in)**2-eye*p2(in)*r2(in)
         aa=conjg(aa1)+aa2
         bb=conjg(bb1)+bb2
         cc=conjg(cc1)+cc2
         phi22=phi22*exp(-bb**2/4.0/aa+cc)
         phi22=phi22*sqrt(-pi/aa)
         if (dreal(aa).gt.0.0) then
            print *,"r1=",r1(in)
            print *,"r1=",p1(in)
            print *,"r1=",g1(in)
            print *,"r2=",r2(in)
            print *,"p2=",p2(in)
            print *,"g2=",g2(in)
            print *,"aa=",aa
            print *,"error"
            stop
         endif
      enddo
      phi22=phi22*conjg(N1)*N2
      if (abs(phi22).gt.1.0E20) phi22=0.0
      if (abs(phi22).lt.1.0E-20) phi22=0.0
      govl=phi22
      return
      end
C-------------------------------------------------------------------------
      FUNCTION gaussian(x,x1,p1,g1)
c
c     Gaussian basis fucntion
c
      IMPLICIT NONE
      INTEGER in
      integer NC,nta,NPT
      PARAMETER(NC=1,NPT=4,nta=100)

      REAL*8 x
      real*8 pi,a1,b1,c1,d1,e1,f1
      complex*16 x1,p1,g1
      COMPLEX*16 EYE,GAU,gaussian,N1
      DIMENSION x(NC),x1(NC),p1(NC),g1(NC)
      dimension a1(NC),b1(NC),c1(NC),d1(NC),e1(NC),f1(NC)
c
      pi=3.141592654
      EYE=(0.0,1.0)
c
      do in=1,NC
         a1(in)=dreal(g1(in))
         b1(in)=dimag(g1(in))
         c1(in)=dreal(x1(in))
         d1(in)=dimag(x1(in))
         e1(in)=dreal(p1(in))
         f1(in)=dimag(p1(in))
      enddo
c
      N1=1.0
```

```
      do in=1,NC
         N1=N1*(a1(in)/pi)**0.25*exp(-0.5*a1(in)*d1(in)**2
     &        -d1(in)*e1(in)-(b1(in)*d1(in)+f1(in))**2/2.0/a1(in))
      enddo
c
      GAU=1.0
      DO in=1,NC
         GAU=GAU*EXP(-0.5*g1(in)*(x(in)-x1(in))**2
     &        +EYE*p1(in)*(x(in)-x1(in)))
      END DO
      GAU=GAU*N1
c
      if (abs(gau).gt.1.0E20) gau=0.0
      if (abs(gau).lt.1.0E-20) gau=0.0
      gaussian=gau
c
      RETURN
      END
C-----------------------------------------------------------------------
      FUNCTION gaussian_type2(x,x1,p1,g1,dt,m)
c
c     Gaussian basis function operated by the kinetic operator
c
      IMPLICIT NONE
      INTEGER in
      integer NC,nta,NPT
      PARAMETER(NC=1,NPT=4,nta=100)
      REAL*8 x,pi,m,dt
      real*8 a1,b1,c1,d1,e1,f1
      complex*16 x1,p1,g1
      COMPLEX*16 EYE,GAU2,rnum,rden,gaussian_type2,N1
      DIMENSION x(NC),x1(NC),p1(NC),g1(NC),m(NC)
      dimension a1(NC),b1(NC),c1(NC),d1(NC),e1(NC),f1(NC)
c
      pi=3.141592654
      EYE=(0.0,1.0)
c
      do in=1,NC
         a1(in)=dreal(g1(in))
         b1(in)=dimag(g1(in))
         c1(in)=dreal(x1(in))
         d1(in)=dimag(x1(in))
         e1(in)=dreal(p1(in))
         f1(in)=dimag(p1(in))
      enddo
c
      N1=1.0
      do in=1,NC
         N1=N1*(a1(in)/pi)**0.25*exp(-0.5*a1(in)*d1(in)**2
     &        -d1(in)*e1(in)-(b1(in)*d1(in)+f1(in))**2/2.0/a1(in))
      enddo
c
      GAU2=1.0
      DO in=1,NC
         rnum=p1(in)/g1(in)-EYE*(x1(in)-x(in))
         rden=2.0/g1(in)+EYE*2.0*dt/m(in)
         GAU2=GAU2*EXP(rnum**2/rden-p1(in)**2/(2.0*g1(in)))
     &        *sqrt(m(in)/(m(in)+eye*g1(in)*dt))
      END DO
      GAU2=GAU2*N1
      if (abs(gau2).gt.1.0E20) gau2=0.0
```

```
      if (abs(gau2).lt.1.0E-20) gau2=0.0
      gaussian_type2=gau2
      RETURN
      END
c-------------------------------------------------------------------------
```

Problem 15:

The solution to this problem can be obtained from
http://xbeams.chem.yale.edu/∼batista/P15/P15.tar
and requires the math libraries from
http://xbeams.chem.yale.edu/∼batista/m.tar.
Untar both files by typing

```
tar -xvf P15.tar
```

and

```
tar -xvf m.tar
```

Type

```
cd P15
```

By typing

```
ls
```

you will see that the problem is solved in terms of MP/SOFT and SOFT simulations in 1d and 4d, allowing for direct comparisons between grid-based calculations and MP/SOFT. In addition, the problem is solved in 24d according to the MP/SOFT method.

To start, type

```
cd P15_1dg
```

Compile the grid-based 1d-version of the program by typing

```
comp_15_1dg
```

Run the program by typing

```
Problem15_g
```

Compute the spectrum by compiling the program calcspec.f by typing

```
./comp_calcspec
```

and running the program by typing

```
./calcspecs
```

Visualize the photoabsorption spectrum by typing

```
gnuplot<peV
```

or

```
gnuplot< pw
```

Analogously, compile the MP/SOFT version of the program in the directory P15_1dmp, with the corresponding script by typing

```
comp_15_1d
```

and run it by typing

```
poe Problem15 -procs 6
```

The output will be the autocorrelation function as a function of time saved in file named autocorr. Results can be compared to reference calculations, stored in file named autocorr_ref.

The photoabsorption spectrum can be obtained by compiling calcspec.f by typing

```
comp_calcspecs
```

and running it by typing

```
calcspec
```

In order to visualize the spectrum, type

```
gnuplot <peV
```

or

```
gnuplot <peV
```

Simulations for the 4-dimensional and 24-d model Hamiltonians can be performed analogously. However, for the current implementation, the 24-d simulations required forward and backward propagation in order to obtain the correlation function as $C(2t) = \langle \Psi(-t) | \Psi(t) \rangle$.

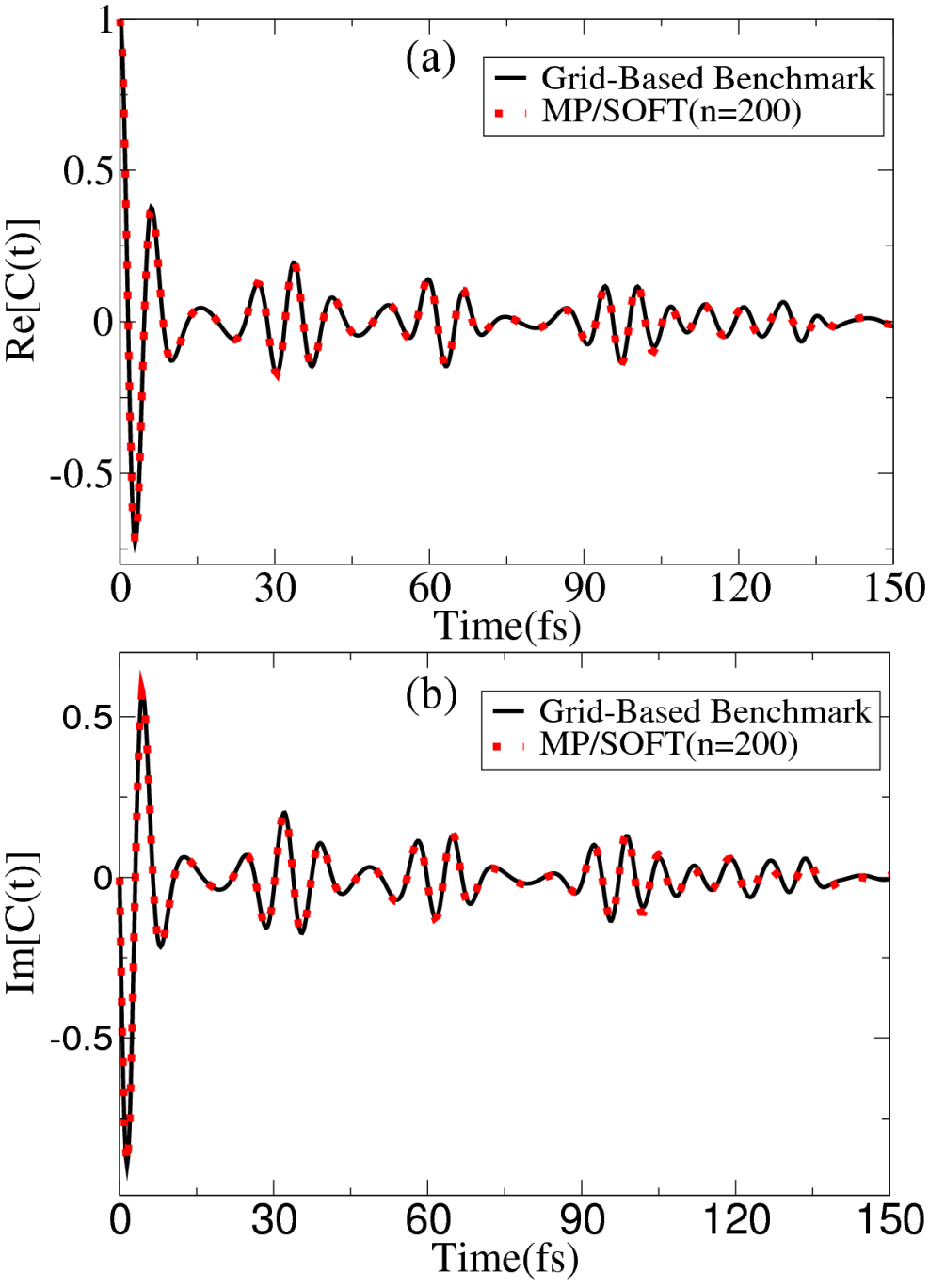

(a)
Grid-Based Benchmark
MP/SOFT(n=200)
Re[C(t)]
1
0.5
0
-0.5
0
30
60
90
120
150
Time(fs)
(b)
Grid-Based Benchmark
MP/SOFT(n=200)
Im[C(t)]
0.5
0
-0.5
0
30
60
90
120
150
Time(fs)

Figure 1:

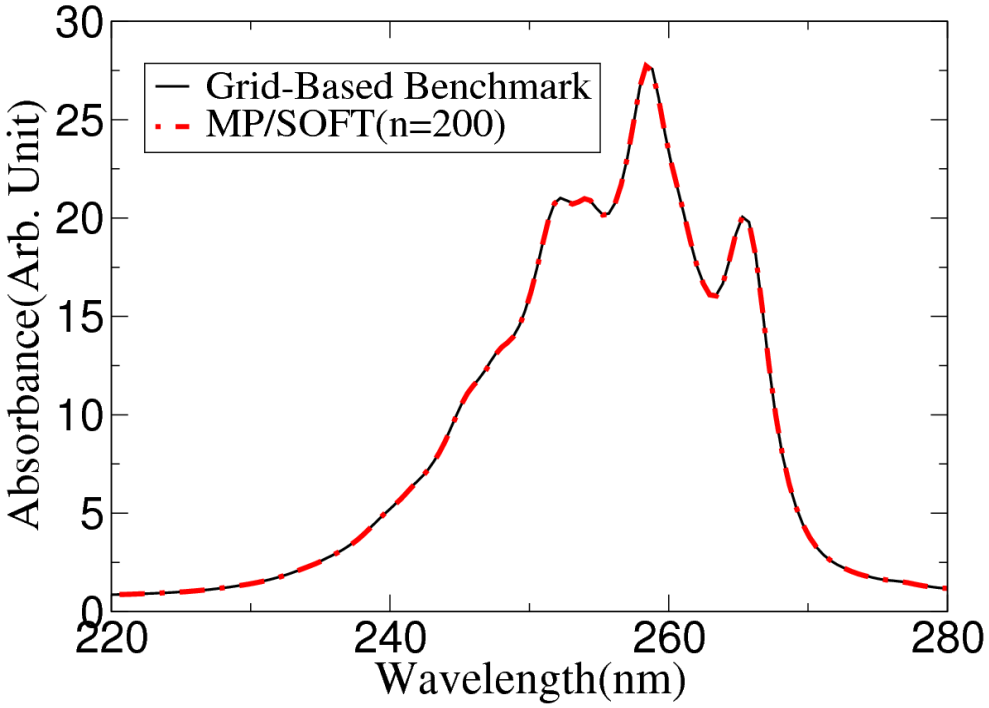

Grid-Based Benchmark
MP/SOFT(n=200)
Absorbance(Arb. Unit)
Wavelength(nm)
0
5
10
15
20
25
30
220
240
260
280

Figure 2:

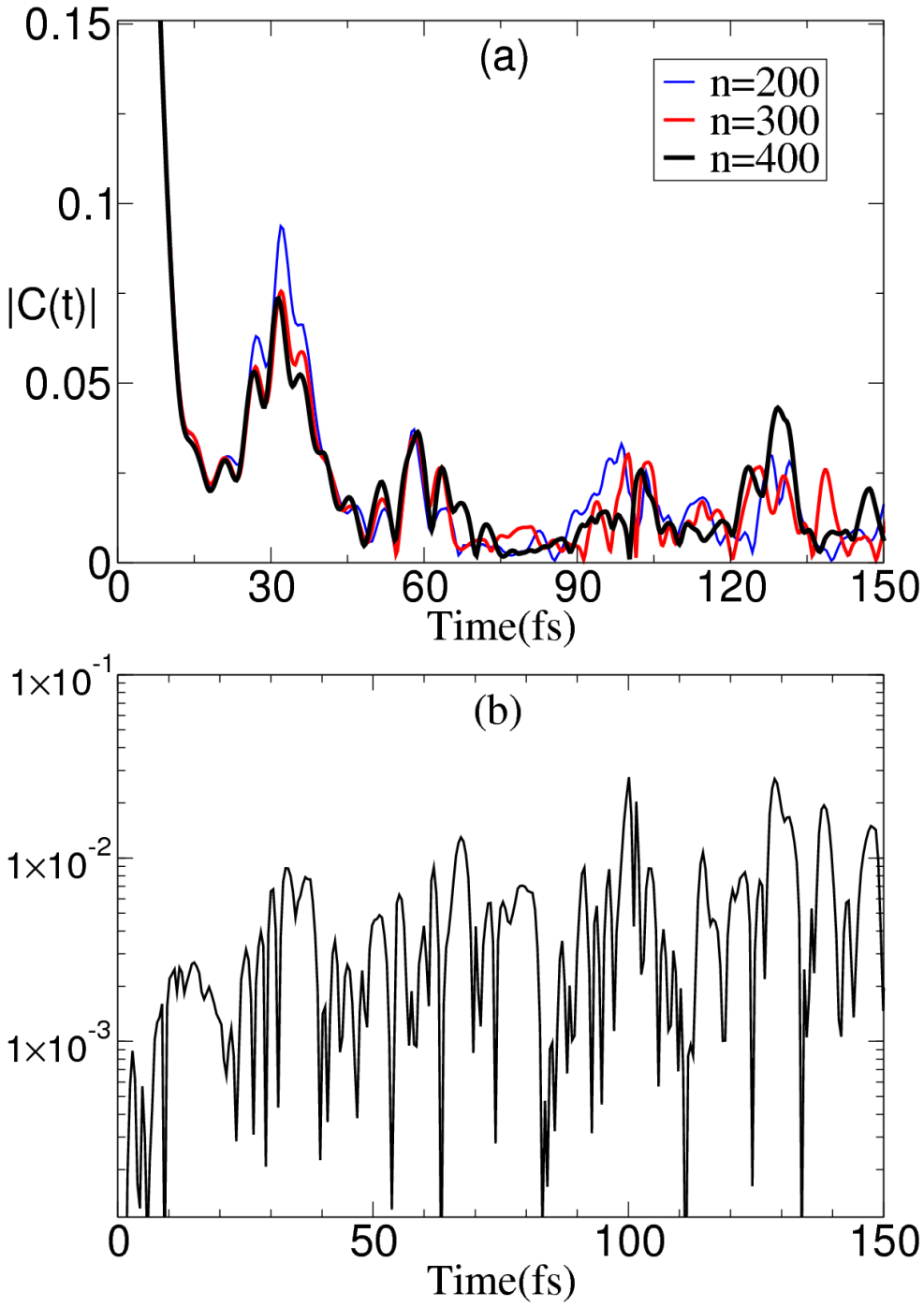

(a)
n=200
n=300
n=400
|C(t)|
0.15
0.1
0.05
0
0
30
60
90
120
150
Time(fs)
(b)
1×10^-1
1×10^-2
1×10^-3
0
50
100
150
Time(fs)

Figure 3:

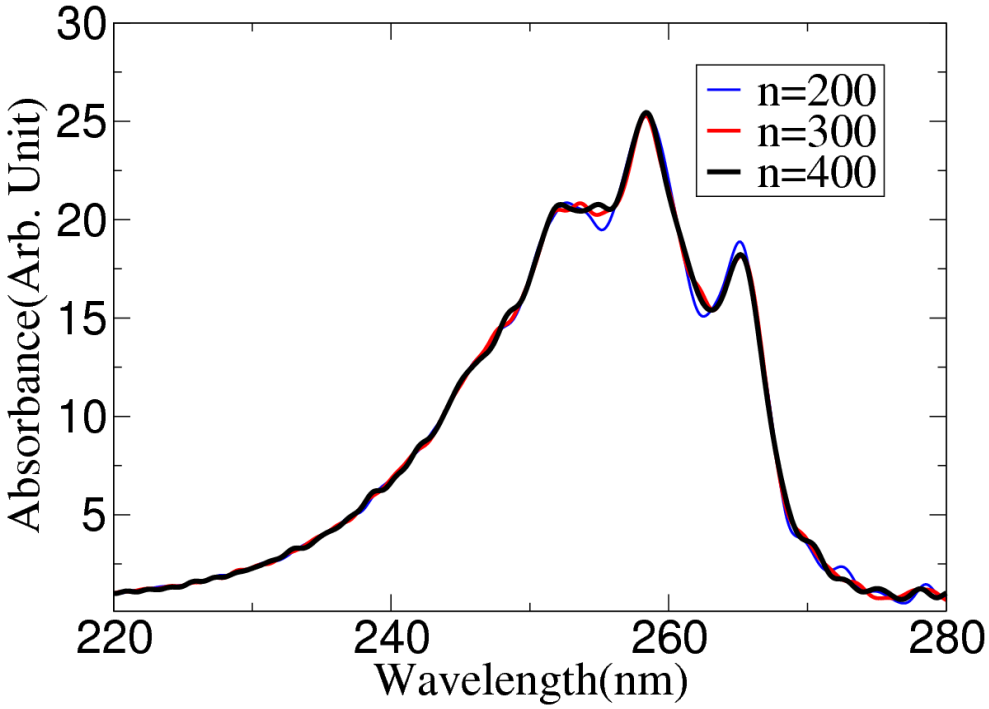

30
25
20
15
10
5
Absorbance(Arb. Unit)
220
240
260
280
Wavelength(nm)
n=200
n=300
n=400

Figure 4:

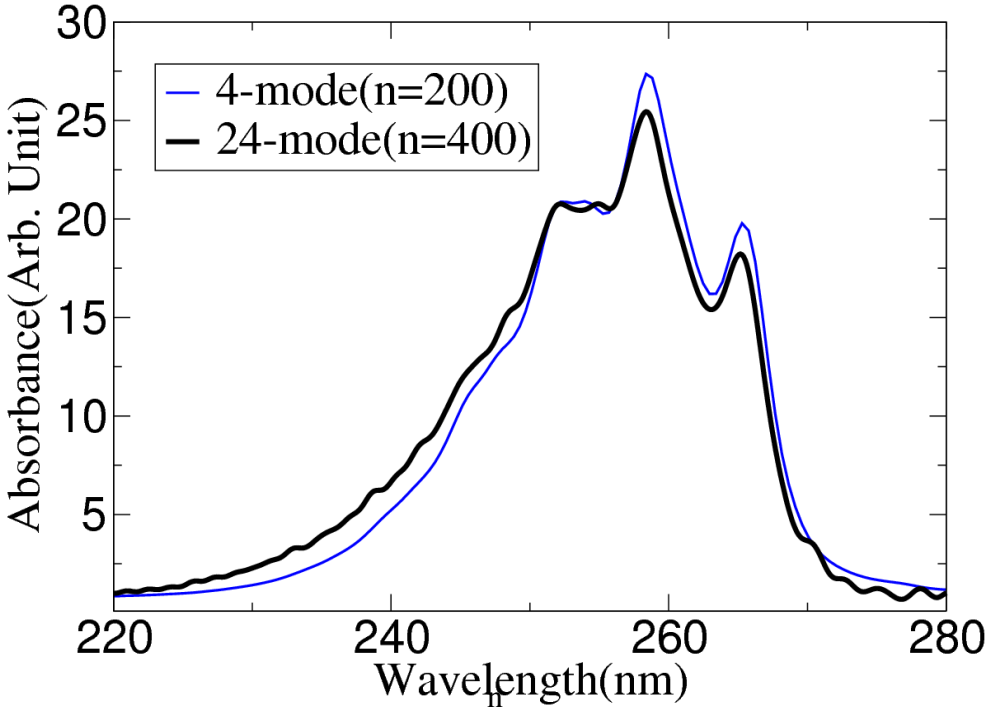
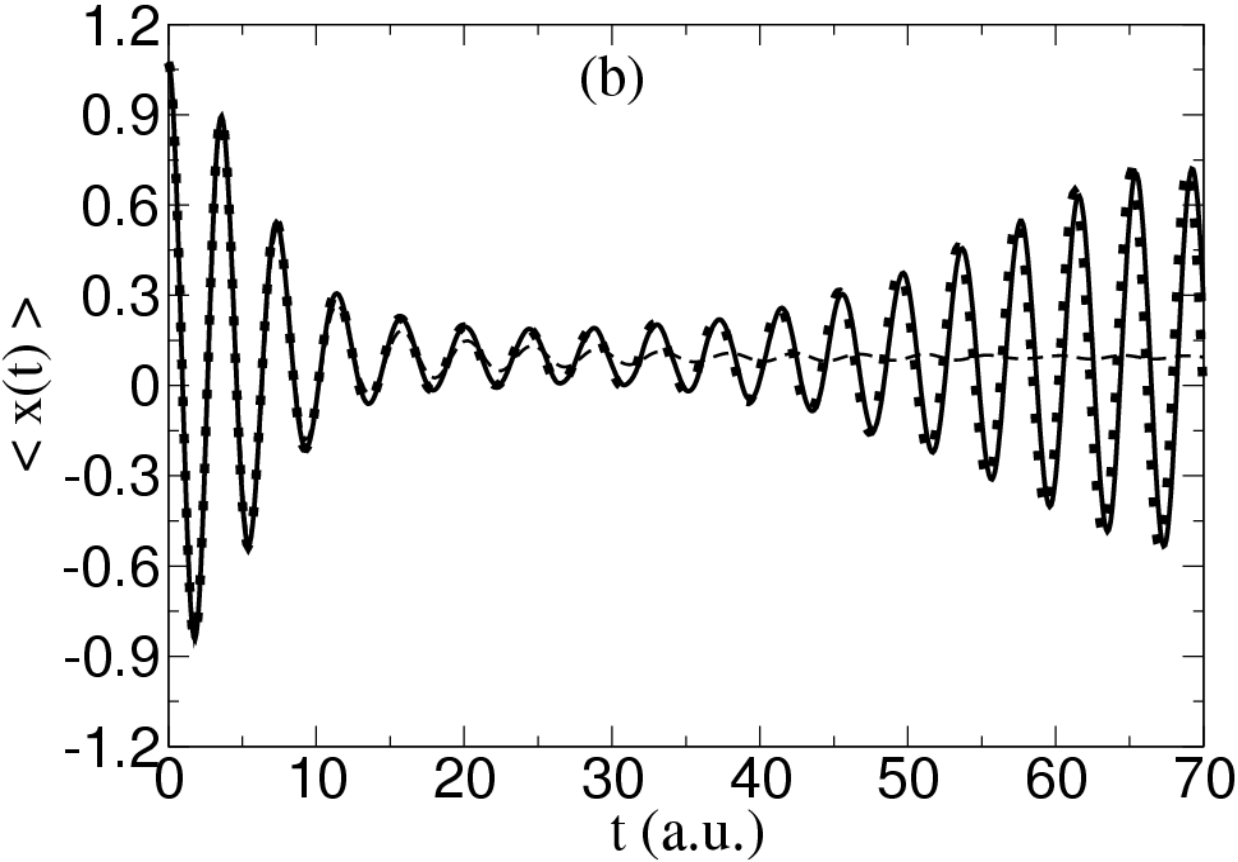
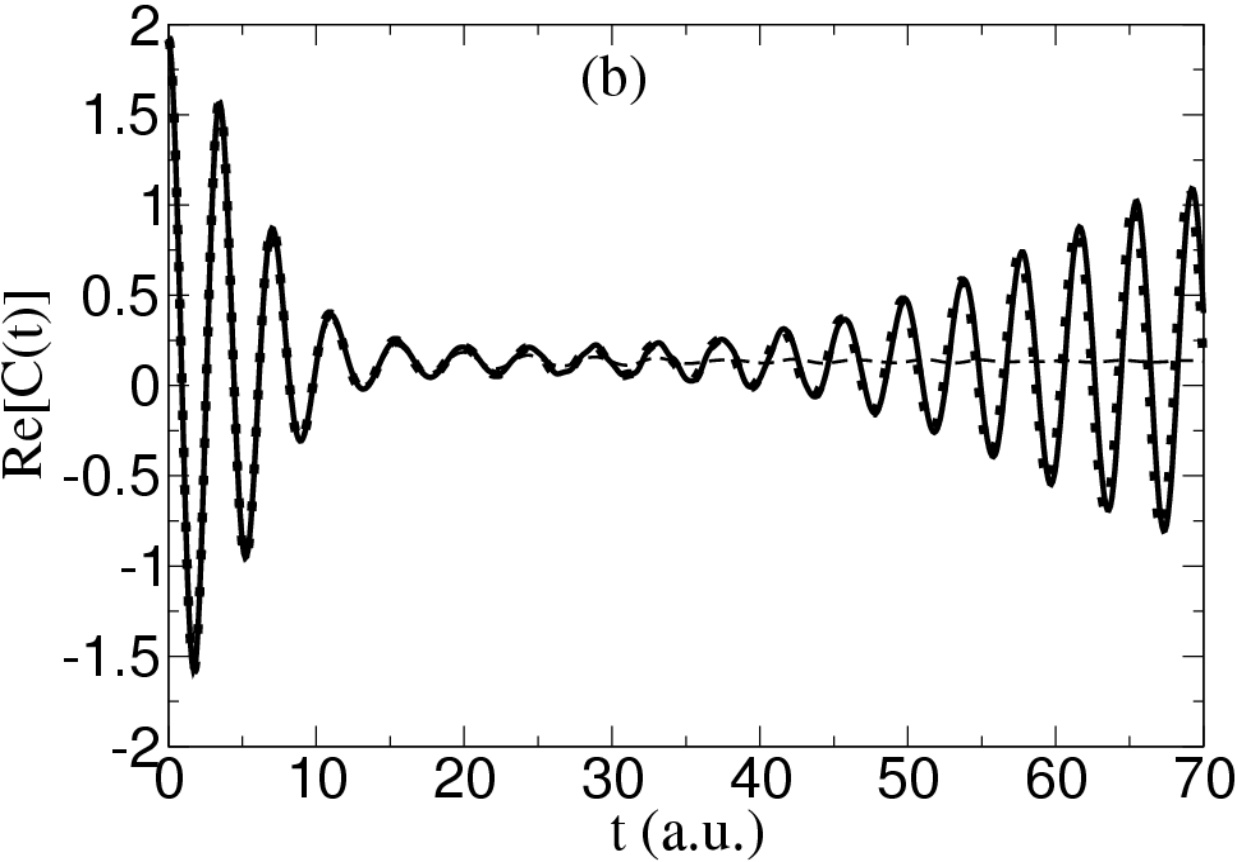

4-mode(n=200)
24-mode(n=400)
Absorbance(Arb. Unit)
Wavelength(nm)
30
25
20
15
10
5
220
240
260
280

Figure 5:

Problem 16:

The solution to this problem can be obtained from
http://xbeams.chem.yale.edu/∼batista/P16/P16.tar
Instructions for compiling and running can be obtained upon request.


(b)
1.2
0.9
0.6
0.3
0
-0.3
-0.6
-0.9
-1.2
< x(t) >
0
10
20
30
40
50
60
70
t (a.u.)


(b)
Re[C(t)]
t (a.u.)
2
1.5
1
0.5
0
-0.5
-1
-1.5
-2
0
10
20
30
40
50
60
70