Optimal control of the quantum harmonic oscillator

Greg von Winckel

July 22, 2014

Suppose we have an electron confined in a quantum harmonic oscillator and subjected to a spatially-constant time-varying electric field of our choosing. Can we find an $\mathcal{E}(t)$ so that a particle in state $|i\rangle$ at $t=0$ has maximimal probability of being in state $|f\rangle$ at some $t=T$? This can be posed as a constrained optimization problem

$$\min_{\psi,\mathcal{E}} -|\langle\psi(T)|f\rangle|^2 + \frac{\gamma}{2}\|\mathcal{E}\|^2 \tag{1}$$

where γ is a small regularization parameter. subject to the constraint

$$i\hbar\dot{\psi} = \left\{-\frac{\hbar^2}{2m} + \frac{1}{2}m\omega^2x^2 + \mathcal{E}(t)x\right\}\psi \tag{2}$$

We can rescale to obtain

$$i\dot{\psi} = \left\{-\frac{1}{2}\partial_x^2 + \frac{1}{2}x^2 + u(t)x\right\}\psi \tag{3}$$

1 Optimize Before Discretize (OBD)

One approach to solve the optimal control problem is work in function space and take Gâteaux derivatives to obtain a system of optimality equations which are then discretized. If we wish to be precise, we can say that $\psi : W \to \mathbb{C}$ where $W = \Omega \times I$. $I = [0,T]$ and $\Omega = \mathbb{R}$ and that

$$\psi \in \Psi = \{\varphi \in L^2(0,T;H^1(\Omega))|\varphi(x,0) = \psi_0(x)\} \tag{4}$$

We can also specify a control space $u \in U = L^2(I)$.

1.1 Method of Lagrange Multipliers

Let $\lambda(x,t)$ be the Lagrange multiplier for the equality constraint. We can form a Lagrangian

$$\mathcal{L}(\psi,y,u) = -(\bar{\psi}(\cdot,T),\mathcal{P}_f\psi(\cdot,T))_\Omega + \frac{\gamma}{2}\|u\|_{L^2}^2 - (\lambda,[i\partial_t - \mathcal{H}(u)]\psi)_W - (\bar{\lambda},[-i\partial_t - \mathcal{H}(u)]\bar{\psi})_W \tag{5}$$

where $\mathcal{P}_f$ is the orthogonal projector onto the target state $|f\rangle$ and $\mathcal{H}(u)$ is the term in braces in (3). First taking the Gâteaux derivative of the Lagrangian with respect to ψ

$$(\mathcal{L}_\psi, \delta\psi)_W = \frac{d}{d\tau}\mathcal{L}(\psi + \tau\delta\psi, u, \lambda)\big|_{\tau=0} \tag{6}$$

Since the Lagrangian is linear in ψ, evaluating is quite straightforward

$$(\mathcal{L}_\psi, \delta\psi)_W = -(\bar{\psi}, \mathcal{P}_f\delta\psi)_\Omega\big|_{t=T} - (\lambda, [i\partial_t - \mathcal{H}(u)]\delta\psi)_W \tag{7}$$

where $\psi + \tau\delta\psi \in \Psi$ for all τ, so $\delta\psi$ must satisfy homogeneous initial conditions. We can denote this as $\delta\psi \in \Psi_H$, which is a homogeneous initial condition analog of Ψ.

Integrating by parts,

$$(\lambda, \partial_t\delta\psi)_W = -(\partial_t\lambda, \delta\psi)_W + (\lambda, \delta\psi)_\Omega\big|_{t=0}^{T} \tag{8}$$

So we can write the action of the ψ-derivative of the Lagrangian on a test function as

$$(\mathcal{L}_\psi, \delta\psi)_W = -(\partial_t\lambda, \delta\psi)_W - i(\lambda, \delta\psi)_\Omega\big|_{t=0}^{T} + ([i\partial_t + \mathcal{H}(u)]\lambda, \delta\psi)_W \tag{9}$$

Two things that are worth mentioning about this equation is that $\mathcal{H}(u)$ is self-adjoint on since there can be no surface terms from integating by parts in space as any element of Ψ vanishes at the "boundaries" of $\pm\infty$. Not only that, but they must have asymptotic decay faster than x^{-2}. The second point is that since $\delta\psi \in \Psi_H$, the surface term $(\lambda, \delta\psi)_\Omega\big|_{t=0} = 0$. We can then split (9) into a volume term and surface terms. A necessary condition for a minimizer is that both be equal to zero. The volume term is

$$([i\delta_t + \mathcal{H}(u)]\lambda, \delta\psi) = 0, \quad \forall\delta\psi \in \Psi_H \tag{10}$$

and the surface terms are

$$-(\mathcal{P}_f\bar{\psi}, \delta\psi)_\Omega\big|_{t=T} - i(\lambda, \delta\psi)_\Omega\big|_{t=T} = 0, \quad \forall\delta\psi \in \Psi_H \tag{11}$$

From this, we obtain the adjoint equation

$$-i\dot{\lambda} = \mathcal{H}(u)\lambda, \quad \lambda(x,T) = i\mathcal{P}_f\bar{\psi}(x,T) \tag{12}$$

Next we take variations with respect to the control

$$(\mathcal{L}_u, \delta u)_W = \frac{d}{d\tau}\mathcal{L}(\psi, u + \tau\delta u, \lambda)\big|_{\tau=0} \tag{13}$$

This is also fairly simple to carry out.

$$(\mathcal{L}_u, \delta u)_W = \gamma(u, \delta u)_I + (\lambda, \mathcal{H}'(u)\delta u\psi)_W + (\bar{\lambda}, \mathcal{H}'(u)\delta u\bar{\psi})_W \tag{14}$$

In our case of a dipole driven system

$$\mathcal{H}'(u) = x \tag{15}$$

We can rewrite this as

$$(\mathcal{L}_u, \delta u)_W = \gamma(u, \delta u)_I + ((\lambda, x\psi)_\Omega, \delta u)_I + ((\bar{\lambda}, x\bar{\psi})_\Omega, \delta u)_I \tag{16}$$

1.2 Discretization

Requiring the above to be zero for all $\delta u \in L^2(I)$ gives us the reduced gradient

$$\nabla \tilde{J}(u) = \frac{d}{du} J(y,u) = \mathcal{L}_u = \gamma u + 2\mathrm{Re}[(\lambda, x\psi)_\Omega] \tag{17}$$

Expand the time-dependent solutions in terms of the eigenfunctions

$$\psi(x,t) = \sum_{k=0}^{\infty} y_k(t)\phi_k(x), \quad \lambda(x,t) = \sum_{k=0}^{\infty} z_k(t)\phi_(x) \tag{18}$$

$$\phi_k(x) = \frac{1}{\sqrt{2^k k! \sqrt{\pi}}} H_k(x) e^{-x^2/2} \tag{19}$$

Where $H_k(x)$ is the kth Hermite polynomial. In practice, we truncate the series after m terms. This basis gives rise to the matrices

$$(\phi_j, \phi_k) = \delta_{jk} \tag{20}$$

$$\frac{1}{2}(\phi_j', \phi_k') + \frac{1}{2}(\phi_j, x^2\phi_k) = \frac{j+1}{2}\delta_{jk} \tag{21}$$

$$(\phi_j, x\phi_k) = \begin{cases} \frac{\sqrt{j+1}}{2} & k = j+1 \\ \frac{\sqrt{k+1}}{2} & k = j-1 \end{cases} \tag{22}$$

The semidiscrete form of the state and adjoint equations are then

$$\frac{d\mathbf{y}}{dt} = -i[\mathbf{H} + u(t)\mathbf{X}]\mathbf{y}, \quad \frac{d\mathbf{z}}{dt} = i[\mathbf{H} + u(t)\mathbf{X}]\mathbf{z} \tag{23}$$

Discretizing the state equation in time with the implicit midpoint rule

$$\mathbf{y}^n = \mathbf{y}^{n-1} - i\Delta t[\mathbf{H} + u(t_{n-1/2})\mathbf{X}]\left(\frac{\mathbf{y}^n + \mathbf{y}^{n-1}}{2}\right) \tag{24}$$

Rearranging this, we have the update formula

$$\left(\mathbf{I} + \frac{i\Delta t}{2}[\mathbf{H} + u_n\mathbf{X}]\right)\mathbf{y}^n = \left(\mathbf{I} - \frac{i\Delta t}{2}[\mathbf{H} + u_n\mathbf{X}]\right)\mathbf{y}^{n-1}, \quad n = 1, \ldots, N \tag{25}$$

where $u_n \equiv u(t_{n-1/2})$ with the initial condition $\mathbf{y}(0) = \hat{e}_i$.

We obtain a similar formula for the fully discretized adjoint equation

$$\left(\mathbf{I} + \frac{i\Delta t}{2}[\mathbf{H} + u_n\mathbf{X}]\right)\mathbf{z}^{n-1} = \left(\mathbf{I} - \frac{i\Delta t}{2}[\mathbf{H} + u_n\mathbf{X}]\right)\mathbf{z}^n, \quad n = 1, \ldots, N \tag{26}$$

with the "final condition" that

$$\mathbf{z}^N = i\mathbf{P}_f\bar{\mathbf{y}}^N \tag{27}$$

where $\mathbf{P}_f$ is a discrete form of the dyadic operator $\mathcal{P}_f$ and will consist of all zeros except a 1 on the diagonal element corresponding to state f. The kth component of the discretized reduced gradient is

$$[\nabla \tilde{J}(u)]_k = \gamma\Delta t u_k + \frac{\Delta t}{2}\mathrm{Re}[(z^k + z^{k-1}) \cdot \boldsymbol{X}(\boldsymbol{y}^k + \boldsymbol{y}^{k-1}] \tag{28}$$

Where did this formula come from? Well, we have to approximate the function space inner products in (16) numerically. Since we only know the control on the midpoints and the control has less regularity than the state and adoint variables, we don't want to interpolate it. Instead, we interpolate $\boldsymbol{y}(t)$ and $\boldsymbol{z}(t)$ to the midpoint and then use the midpoint rule for numerical integration, which amounts to multiplication by Δt.

Two points to remember about OBD are that the Lagrange multiplier satisfies exactly the same type of equation as the state variable, except backward in time and we obtain discretized multipliers at $N+1$ points in time.

The OBD method is convenient as we can apply any discretization we like without having to rederive the equations. The potential downside is that the discretized gradient is an approximation to the true gradient and is limited by the quality of the approximation. There is no general guarantee that the Strong Wolfe conditions for finding a local linesearch minimizer can be satisfied[1].

1.3 Implicit function approach

It is not required to use Lagrange multipliers to solve the optimal control problem. Instead, we can simply treat the state ψ as an implicit function of the control u.

$$\tilde{J}(u) = J(\psi(u), u) = \frac{\gamma}{2}\|u\|^2 - (\bar{\psi}, \mathcal{P}_f\psi)_\Omega\big|_{t=T} \tag{29}$$

Where the dependence of ψ on u is defined implicitly by the equality constraint

$$e(\psi, u) = i\partial_t\psi - \mathcal{H}(u)\psi = 0 \tag{30}$$

The reduced gradient is

$$\nabla\tilde{J}(u) = \frac{d}{du}J(\psi(u), u) = \frac{\partial J}{\partial u} + \frac{\partial J}{\partial \psi}\frac{\partial \psi}{\partial u} + \frac{\partial J}{\partial \bar{\psi}}\frac{\partial \bar{\psi}}{\partial u} \tag{31}$$

As before, we shall keep things scalar by looking at the directional derivative of $\tilde{J}$ evaluated at u in a direction δu

$$(\tilde{J}_u, \delta u) = \gamma(u, \delta u)_I - (\bar{\psi}, \mathcal{P}_f\delta\psi)_\Omega - (\psi, \mathcal{P}_f\bar{\psi})_\Omega \tag{32}$$

So what is $\delta\psi$? This is going to be the differential change in the state due to a differential change in the control δu

$$\delta\psi = \frac{\partial\psi}{\partial u}\delta u \tag{33}$$

Using the implicit function theorem, we find the plane tangent to the equality constraint

$$e_\psi\delta\psi + e_u\delta u = 0 \tag{34}$$

[1] I think, but haven't tried to prove that symmetric Galerkin methods in space and the implicit midpoint rule in time lead to the same discretized optimality system for the linear Schrödinger equation regardless of whether we optimize or discretize first, but this is a special case.

Or we could say that

$$\delta\psi = -e_\psi^{-1} e_u \delta u \tag{35}$$

Here e_ψ and e_u are linear operators.

$$e_\psi = i\partial_t - \mathcal{H}(u), \quad e_u = -\mathcal{H}'(u)\psi = x\psi \tag{36}$$

This means that if you give a differential change in control, δu you can solve for the differential change in state by solving the PDE

$$[i\partial_t - \mathcal{H}(u)]\delta\psi = x\psi\delta u \tag{37}$$

with the homogeneous initial condition stipulated by $\delta\psi \in \Psi_H$.
Substituting this expression into (32), gives us a term of the form

$$(\bar{\psi}, \mathcal{P}_f \delta\psi)_W = (\bar{\psi}, \mathcal{P}_f [i\partial - \mathcal{H}(u)]^{-1}[x\psi\delta u])_W = ([i\partial - \mathcal{H}(u)]^{-1}\mathcal{P}_f\bar{\psi}, x\psi\delta u)_W \tag{38}$$

where $[i\partial - \mathcal{H}(u)]^{-1}\mathcal{P}_f\bar{\psi}$ is essentially what we called λ above.

2 Discretize Before Optimize (DBO)

The DBO method involves first making a discrete approximation to the cost functional and the state equation and taking variations in a finite dimensional space to obtain a vector-valued gradient. The advantage here is that the gradient will be exact and the Strong Wolfe conditions can be satisfied, however, the equations obtained are discretization specific, which can lead to some inflexibility.

2.1 Discretization

We begin by using the same Hermite Galerkin and implicit midpoint discretizations as before to obtain the fully discrete optimization problem $\min_{y,u} J(y,u)$ where

$$J(y,u) = -\bar{\boldsymbol{y}}^N \cdot \boldsymbol{P}_f \boldsymbol{y}^N + \frac{\gamma\Delta t}{2}\sum_{n=1}^{N} u_n^2 \tag{39}$$

subject to the N equality constraints, where the nth constraint is

$$\mathbf{e}_n(y,u) = \left(\mathbf{I} + \frac{i\Delta t}{2}[\mathbf{H} + u_n\mathbf{X}]\right)\mathbf{y}^n - \left(\mathbf{I} - \frac{i\Delta t}{2}[\mathbf{H} + u_n\mathbf{X}]\right)\mathbf{y}^{n-1} \tag{40}$$

We can denote this in shorthand as

$$\boldsymbol{A}_n = \boldsymbol{I} + \frac{i\Delta t}{2}[\boldsymbol{H} + u_n\boldsymbol{X}], \quad \frac{\partial \boldsymbol{A}_j}{\partial u_k} = \frac{i\Delta t}{2}\boldsymbol{X}\delta_{jk} \tag{41}$$

Using a Lagrange multiplier for every time step equality constraint as well as its complex conjugate.

$$\mathcal{L}(y,u,z) = J(y,u) - \sum_{n=1}^{n} \boldsymbol{z}^n \cdot (\boldsymbol{A}_n\boldsymbol{y}^n - \overline{\boldsymbol{A}}_n\boldsymbol{y}^{n-1}) + \overline{\boldsymbol{z}}^n \cdot (\overline{\boldsymbol{A}}_n\overline{\boldsymbol{y}}^n - \boldsymbol{A}_n\overline{\boldsymbol{y}}^{n-1}) \tag{42}$$

Note that J also depends on $\bar{y}$, but it is not written explicitly.
Taking variations with respect to $\boldsymbol{y}^j$

$$\frac{\partial \mathcal{L}}{\partial \boldsymbol{y}^j} \cdot \delta \boldsymbol{y}^j = \boldsymbol{z}^j \cdot \boldsymbol{A}_j \delta \boldsymbol{y}^j - \boldsymbol{z}^{j+1} \cdot \overline{\boldsymbol{A}}_{j+1} \delta \boldsymbol{y}^j \tag{43}$$

In the case where $j = N$

$$\frac{\partial \mathcal{L}}{\partial \boldsymbol{y}^N} \cdot \delta \boldsymbol{y}^N = \boldsymbol{z}^N \cdot \boldsymbol{A}_N \delta \boldsymbol{y}^N - \bar{y}_1^N \cdot \boldsymbol{P}_f \delta \boldsymbol{y}^N \tag{44}$$

Taking variations with respect to u_j

$$\frac{\partial \mathcal{L}}{\partial u_j} \delta u_j = \gamma u_j \delta u_j - \Delta t \mathrm{Im}[\boldsymbol{\lambda}^j \cdot \boldsymbol{X}(\boldsymbol{y}^j + \boldsymbol{y}^{j-1})] \delta u_j \tag{45}$$

To compute the cost functional we solve the state equation and to compute the reduced gradient we solve adjoint equation

$$\boldsymbol{A}_j \boldsymbol{\lambda}^j = \overline{\boldsymbol{A}}_{j+1} \boldsymbol{\lambda}^{j+1}, \quad \boldsymbol{A}_N \boldsymbol{\lambda}^N = \bar{y}_1^N \hat{e}_1 \tag{46}$$

Using the state and adjoint variable, we can evaluate the reduced gradient

$$\frac{\partial \tilde{J}}{\partial u_j} = \gamma u_j - \Delta t \mathrm{Im}[\boldsymbol{\lambda}^j \cdot \boldsymbol{X}(\boldsymbol{y}^j + \boldsymbol{y}^{j-1})] \tag{47}$$

Eliminating matrix multiplication

We can remove the banded matrix multiplication from the right-hand-side in the state and adjoint equations by observing that our time stepping formula has the general form (diagonal plus/minus banded)

$$(D + B)y^k = (D - B)y^{k-1} \tag{48}$$

Let $s = y^k + y^{k-1}$ so we have

$$(D + B)(s - y^{k-1}) = (D - B)y^{k-1} \tag{49}$$

Since y^{k-1} is known, all terms of this type can be moved to the right-hand-side

$$(D + B)s = 2Dy^{k-1}, \quad y^k = s - y^{k-1} \tag{50}$$

In the case of the DBO adjoint equation, it is slighltly different as the banded matrix on the left and right are multiplied by (in general) different control amplitudes. We have something of the form

$$(D + u_k B)z^k = (D - u_{k+1} B)z^{k+1} \tag{51}$$

We make the Ansatz

$$s = z^{k+1} + \alpha z^k \tag{52}$$

and substitute this into the update formula

$$(D + u_k B)s = (D + \alpha I)z^{k+1} + (\alpha u_k - u_{k+1})Bz^{k+1}, \quad z^k = z - \alpha z^{k+1} \tag{53}$$

The banded term on the right-hand-side can be eliminated by setting $\alpha = u_{k+1}/u_k$, however, care must be taken if this ratio becomes large or undefined since we cannot reasonably apply this strategy. This is a casualty of the DBO approach with the implicit midpoint theorem.

Note that this is not a significant savings for our problem anyway, unless the bandwidth would be much larger. The solves are always going to be more expensive than the multiplies.

3 Python Implementation

To implement the controlled harmonic oscillator code in Python, we first observe that there are some quantities which will be used repeatedly, namely the matrices used in time-stepping the forward and adjoint problem. For this reason it makes sense to encapsulate this data in a class which will have external methods for evaluating the cost and gradient and internal methods for solving the forward and adjoint problems, since a black box minimizer does not need to know what is being done for this part.

```
from __future__ import division
import numpy as np
import matplotlib.pyplot as plt
from scipy.linalg import solve_banded
from scipy.optimize import fmin_bfgs, fmin_cg
from mult_banded import mult_banded
```

Our class should contain the following member data

1. Number of time steps
2. Time step size
3. The initial given and fine state desired
4. Number of Hermite functions to use
5. All of the matrices we will use stored in banded form
6. The control, state, and adjoint variables

```
class QHO(object):
    """
    Quantum Harmonic Oscillator class
    """
    def __init__(self,modes,N,T,i,f,g):
        """
        Create and store the three banded matrices for the system
        (I,H, and X) since they will be reused
        """
        self.modes = modes
        self.N = N
        self.dt = T/N
        self.i = i
        self.f = f
        self.g = g

        self.u = np.zeros(N)

        j = np.arange(modes)

        # Identity matrix in banded form
        self.I = np.zeros((3,modes))
        self.I[1,:] = 1

        # Stationary part of Hamiltonian in banded form
        self.H = np.zeros((3,modes))
        self.H[1,:] = j+0.5

        # Dipole term in banded form
        self.X = np.zeros((3,modes))
        x = np.sqrt(j+1)/2
        self.X[0,1:] = x[:-1]
        self.X[2,:-1] = x[:-1]

        # Allocate space for the state variable
        self.Y = np.zeros((modes,N+1),dtype=complex)

        # Allocate space for the adjoint variable
        self.Z = np.zeros((modes,N+1),dtype=complex)
```

In order to evaluate the cost, we must solve the state equation, however, direct access to this method is not needed by a blackbox optimizer, so we will denote this as an internal use method. Of course nothing prevents us from calling this method outside the class.

```
    def _solve_state(self,u):
        """
        Compute the solution to the state equation for a given
        initial condition y[i]=1 and control u. See eq. (25)
        """

        self.Y[self.i,0] = 1

        for k in range(self.N):
            ab = self.I+0.5j*self.dt*(self.H+u[k]*self.X)
            b = mult_banded((1,1),np.conj(ab),self.Y[:,k])
            self.Y[:,k+1] = solve_banded((1,1),ab,b,overwrite_ab=True,
                                         overwrite_b=True,debug=False,
                                         check_finite=True)
        self.u = u
```

The reason we want to store the control as a member variable is that we can check when a new control is given by the blackbox optimizer and recompute the state only when required. The adjoint solver is fairly similar

```
    def _solve_adjoint(self,u):
        """
        Compute the solution to the adjoint equation for a given
        final condition and control u. See eq. (26)
        """

        self.Z[self.f,-1] = 1j*np.conj(self.Y[self.f,-1])

        for j in range(self.N):
            k = self.N-j-1
            ab = self.I+0.5j*self.dt*(self.H+u[k]*self.X)
            b = mult_banded((1,1),np.conj(ab),self.Z[:,k+1])
            self.Z[:,k] = solve_banded((1,1),ab,b,overwrite_ab=True,
                                       overwrite_b=True,debug=False,
                                       check_finite=True)
        self.u = u
```

Now we add a method for evaluating the cost functional

```
    def cost(self,u):
        """
        Evaluate the reduced cost functional
        """

        if not np.array_equal(u,self.u):
            self._solve_state(u)

        J = 0.5*self.dt*self.g*sum(u**2)-abs(self.Y[self.f,-1])**2
        return J
```

Note that we update the state only if we have been given a new control. The implementation of the gradient is

```
    def grad(self,u):
        """
        Evaluate the reduced gradient
        """

        if not np.array_equal(u,self.u):
            self._solve_state(u)

        self._solve_adjoint(u)

        dex = range(1,self.N+1)


        # Inner product term in eq (28)
        ip = [np.dot((self.Z[:,j]+self.Z[:,j-1]),
              mult_banded((1,1),self.X,(self.Y[:,j]+self.Y[:,j-1])))
              for j in dex]

        dJ = self.dt*(self.g*u+0.5*np.real(np.array(ip)))

        return dJ
```

Time to test it out

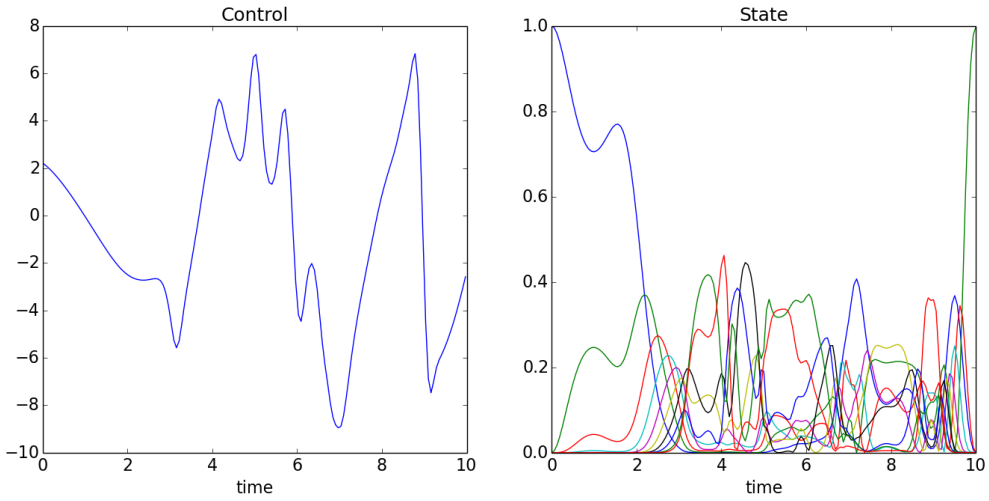
```
if __name__ == '__main__':

    # Number of Hermite functions
    modes = 10

    # Number of time steps
    N = 50

    # Control duration
    T = 5

    # Regularization parameter
    g = 1e-1

    # Time grid
    t = np.linspace(0,T,N+1)

    # Midpoints of time steps
    tm = 0.5*(t[:-1]+t[1:])

    # Initial guess of control
    u = 0.01*np.ones(N)

    # Instantiate Quantum Harmonic Oscillator for 0 -> 1 transition
    qho = QHO(modes,N,T,0,1,g)
```

To compute an optimal control, we start on a coarse time grid with a relatively large regularization parameter and compute a minimizer with the BFGS method. Then we "interpolate" the optimizer onto a grid with twice as many time steps, halve the regularization parameter, and restart with this interpolated optimizer as an initial guess. Experimentation with a fixed grid and regularization indicates that it is difficult both compute a control which is relatively "nice" in the sense of smoothness and amplitude so that we can believe the numerical approximation to the state is converged and to drive the state close to the target.

The computed control does indeed drive the transition $|0\rangle \to |1\rangle$ where the state blue curve is the probability of state $|0\rangle$ being occupied as a function of time and the green curve is the probability of state $|1\rangle$ being occupied.

There are some ways in which this is not a very satisfying result due to the amount of state oscillation, however, and their are questions to be considered. If we perturb the control slightly, does the state change slightly or a lot? Is the BFGS method "skipping over" local minimizers which could lead to a less oscillilatory solution and have a smaller amplitude control? Not having access to the data contained in the line search or trust region method used by `fmin_bfgs` leaves us only to speculate.

This work will be continued with a look at some things we can do to address these points.


Control
8
6
4
2
0
−2
−4
−6
−8
−10
0
2
4
6
8
10
time
State
1.0
0.8
0.6
0.4
0.2
0.0
0
2
4
6
8
10
time