On the optimum control of quantum systems

Erik Natvig

June 8, 2009

Paper in MAT292
Revised version
Department of Mathematics, University of Bergen

Abstract

An important problem in quantum physics is to find control functions for the intensity of electrical fields that change the state of a particle to a desired target state. To do this one can use piecewise constant control functions and use optimization to find the constant values of this that give a result that is as close as possible to the desired state. We represent the "closeness" as an objective function, and to find this we solve the differential equation that governs the particle using the matrix exponential. The resulting objective function is difficult to optimize efficiently using the standard optimization methods. So instead we apply a one-parameter line search method due to Vadim Krotov. It relies on a low-order approximation of the derivative of the objective function with the respect one optimization variable at a time. Thus we need the derivative of the matrix exponential. This is not trivial since the matrix depends on the control function, so we use a "dexp-expansion". A control function with constants that tend to infinity can also give high yield, but this is not desirable, so a term that penalizes erratic controls must be added to the objective function. But if this regularization term term is too large, we instead get close to zero controls and low yield. Careful consideration of both the width of the intervals with constant controls and the size of the regularization term is essential to find a good result, and also the physical parameters must be taken into consideration.

1 Introduction

The complex two-dimensional vector function $\Psi(t)$ represents a certain state of a particle at time t. In for example quantum computing and magnetic resonance imaging, one needs to control the state of particles, and this can be done using an electrical field of varying intensity. We wish to move the particle from one state, Ψ_0, at time 0, to a target state Ψ_T at time T. Under the influence of an electrical field with intensity $J(t)$ that varies with time, the state of the particle is governed by the following differential equation:

$$\Psi' = -i \begin{bmatrix} J(t) & B \\ B & 0 \end{bmatrix} \Psi, \Psi(0) = \Psi_0.$$

Here, B is a physical property which in our time scale can be considered to be constant. We want to find the function $J(t)$ that 'moves' the particle to our desired target state. However, in most cases from practical applications, it is very difficult or even impossible to find such functions analytically.

In this paper I will attempt to alleviate this problem by defining $J(t)$ as a piecewise constant function, where $J(t) = c_{k+1}$ for $t_k \leq t \leq t_{k+1}$. We wish to find the values c_k that make $\Psi(T)$ as close as possible to our desired target Ψ_T. In order to do this, we define a functional $\mathcal{I}(c_1, c_2, ..., c_N)$ that has a high value when $\Psi(T)$ and Ψ_T are close, and use the optimization technique called *line search* to find the values of c_k that maximize $\mathcal{I}$.

I will start in Section 2 with an overview of the subject of optimization and line search. In Section 3 I set up the optimization problem, and continue in Section 4 by presenting the theory needed to maximize the objective functional. Section 5 will deal with applying line search to the problem, and in Section 6 I will show the results of the optimization problem using different parameters. Also I will discuss the results, and in particular show why this kind of optimization is not straight-forward but requires careful consideration of methods and parameters in order to be solved efficiently. Section 7 will conclude the paper.

2 An introduction to optimization

This section deals with general theory of unconstrained optimization. The aim of optimization is to find an extremal point (usually a minimum) of a function f, called the *objective function*, which in the most general case is a mapping from a real-valued vector in $\mathbb{R}^n$ to a scalar in $\mathbb{R}$. So f depends on the entries of a vector x in $\mathbb{R}^n$. The problem is *unconstrained* because x can take any real value. If a maximum of f is desired, usually $-f$ is optimized instead.

Optimization can in a few cases be done analytically and exactly, but if f depends on many variables or is nonlinear, this is usually impossible. Instead, iterative methods must be used, where a starting point and information about the behaviour of the function around this point is used to find the solution iteratively.

2.1 Types of optimization methods

There are two main types of iterative methods for optimization. The type we will not consider here is trust region methods. Here, a model function with behaviour similar to f in a small region around the current point is constructed. Then one looks for a minimum of the model function in this region to find the next point. This is repeated until a minimum is found.

The other type of iterative method is line search, which will be used to solve our problem. The basic idea is to start at a point x_0 in $\mathbb{R}^n$ and choose a search direction p_0. One then looks along that direction for points that give a decrease in function value, and choose a new point x_1 which is better than x_0. At this point, a new search direction p_1 is chosen and a new line search is performed, and this is repeated until a minimum is found.

2.2 Search directions and step length

There are many possible search directions, and which one to use to get fast convergence depends on the behaviour of the function. The most obvious direction is the *steepest descent* direction, which is the opposite direction of the gradient of f at the point. Also, if the step length is sufficiently small, any direction that makes an angle of less than $\pi/2$ radians with $-\nabla f(x)$ gives a decrease in f ([8] page 31). Other search directions that are commonly used are the *Newton direction*, those used by the *quasi-Newton* methods, and those used by the *nonlinear conjugate gradient* methods. After a search direction has been chosen, a new

function is constructed:

$$\phi(\alpha) = f(x_k + \alpha p_k), \alpha > 0.$$

The ideal would be to find the actual value of α that minimizes ϕ, but this is generally too expensive to compute ([8], page 31). Instead, various methods for inexact line search can be used to find an α that gives adequate decrease in ϕ. The step length should not be too short, so that the minimum of f can be found with few iterations. A trade-off between adequate decrease and step length must be made, so that the step length selection algorithm does not spend too much time making a decision.

2.3 Stopping criteria

For any iterative method, it is crucial to have a good strategy to decide when the function value and optimization variables we have obtained are "good enough" to be used as a solution. A solution of an optimization problem should be close enough to an actual extremal point of the objective function. It can be very difficult to choose stopping criteria that work well on a given problem, and usually a combination of the following are used:

- Small change in objective function.
- Objective function starts to increase/decrease.
- Small change in optimization variables.
- Gradient close to the zero vector.

2.4 One-parameter line search

Good knowledge of the optimization problem is crucial in order to choose the quickest optimization method for solving it. For the problem in this paper, the standard search directions such as steepest descent or Newton are slow, as they do not take into consideration the possibility to reuse information and clever ways to calculate the partial derivatives. In this paper we will use a method particulary suited for this problem, namely one-parameter line search (also known as the Krotov Method, see [6]), that was used by Zhu and Rabitz in [10] for quantum control problems. It is a conjugate gradient method because it uses the gradient of the function, and it searches one dimension at a time to find the an improved estimate of a local maximum. The strategy is as follows.

Given a function $F(x)$ that maps $\mathbb{R}^n$ onto $\mathbb{R}$. For a local maximum of F near the starting point $x^{[0]}$, the gradient ∇F must be the zero vector. We need to find the values of the components of the vector x that satisfy this. We look for a zero for the first component of ∇F by differentiating F with respect to x_1 and setting equal to zero:

$$\frac{\partial F}{\partial x_1} = 0.$$

If F is a simple function, this is easy and can be done exactly by using analytical methods. However, this is not the case in this paper, so instead we use a low order (zero'th or first) approximation to $\frac{\partial F}{\partial x_1}$ around the starting point $x_1^{[0]}$, and solve for a new value $x_1^{[1]}$. This is comparable to taking one step of the Newton method for finding a root of a function. This new value gets the first component of the gradient ∇F closer to 0.

Next, the same is done with the second component of ∇F to find the second component of $x^{[1]}$. This is repeated for every component of x and a new iterate $x^{[1]}$ is obtained. This can be used as a starting point for a new iteration. $x^{[1]}$ is hopefully closer than $x^{[0]}$ to the actual minimum or maximum of the function. This procedure should then be repeated until one or several stopping conditions are met. The details about the approximation to $\frac{\partial F}{\partial x_k}$, the starting point and the stopping conditions will be given in sections 3 and 5.

3 Setting up the optimization problem

3.1 Discretizing and solving the equation

We now start setting up the optimization problem. We wish to solve our initial value problem

$$\Psi' = -i \begin{bmatrix} J(t) & B \\ B & 0 \end{bmatrix} \Psi,\ \Psi(0) = \Psi_0 \tag{1}$$

for time T. We divide the interval from the starting point to the endpoint [0,T] into N equally spaced subintervals. So we have $\Delta t = t_{k+1} - t_k = T/N$. We choose $J(t)$ as constant on each interval, that is,

$$J(t) = c_{k+1} \text{ for } t_k \leq t \leq t_{k+1} \text{ for } k = 0, ..., (N-1).$$

So c is a vector in $\mathbb{R}^N$. We will often refer to c as *the controls*. On each subinterval $[t_k, t_{k+1}]$ the coefficient matrix of (1) is constant. We define the matrix

$$M_{k+1} = \begin{bmatrix} c_{k+1} & B \\ B & 0 \end{bmatrix}$$

and can write the discretized differential equation as an initial value problem for each value of k in this manner:

$$\Psi' = -i \begin{bmatrix} c_{k+1} & B \\ B & 0 \end{bmatrix} \Psi = -iM_{k+1}\Psi,\ \Psi(t_k) = \Psi_k. \tag{2}$$

So to solve for time T we need to solve equation (2) on one interval at a time, inserting the solution of the previous equation into the initial condition of the next. In order to solve the equation repeatedly in an efficient manner, we need the *matrix exponential*. Since we want to use one-parameter line search, we also need a low-order approximation to its derivative. This will be detailed in the following subsections.

3.2 Definition of the Matrix Exponential

The exponential function e^x has Taylor expansion

$$\sum_{k=0}^{\infty} \frac{1}{k!} x^k.$$

This expansion can also be used for matrices.

Definition 3.1. *Given a square matrix A. Its exponential is defined as*

$$e^A \equiv \exp(A) = \sum_{k=0}^{\infty} \frac{1}{k!} A^k.$$

The matrix exponential can be used to solve systems of linear differential equations with constant coefficient matrices, due to the fact that the exponential of the coefficient matrix corresponds to the fundamental matrix of the system. See [2] page 417. For a first-order differential equation $y' = ay$ with initial condition $y(0) = y_0$, a solution is $y = y_0 e^{at}$. Correspondingly, the initial value problem

$$x' = Ax,\, x(0) = x_0$$

has exact solution

$$x(t) = e^{At} x_0.$$

A further property of the matrix exponential is that if A is a symmetric matrix, so is e^A.

3.3 The derivative of the matrix exponential

The derivative of $\exp(At)$ with respect to t is simply $A\exp(At)$. However, in our optimization problem, we will be dealing with a problem of the type

$$\frac{\partial}{\partial x}\exp(A(x))$$

where we see that the matrix A depends on a scalar variable x. The question is, how do we calculate this derivative? It is not obvious and it is not simple. To see this we need to differentiate the Taylor expansion of the matrix exponential term by term:

$$\begin{aligned} \exp(A(x)) &= \textstyle\sum_{k=0}^{\infty} \frac{A(x)^k}{k!}, \\ \frac{\partial}{\partial x}\exp(A(x)) &= \frac{\partial}{\partial x} \textstyle\sum_{k=0}^{\infty} \frac{A(x)^k}{k!} \\ &= \textstyle\sum_{k=0}^{\infty} \frac{\partial}{\partial x} \frac{A(x)^k}{k!}. \end{aligned}$$

We see that the power of a matrix appears, and it is defined in the following way:

$$[A(x)]^k = A(x) \cdot A(x) \cdot A(x) \cdots A(x) \text{ (k times)}.$$

The rule for differentiation of a power of a variable does not apply to matrices, because matrices do not in general commute. We must instead use the product rule, which we illustrate with an example with a product of three functions:

$$[f(x)g(x)h(x)]' = f'(x)g(x)h(x) + f(x)g'(x)h(x) + f(x)g(x)h'(x).$$

As we see, each term has the derivative of one function with the rest untouched. We apply the product rule to the power of matrices, using A_x to denote $\frac{\partial}{\partial x}(A(x))$ and A to denote $A(x)$, and get

$$\frac{\partial}{\partial x}\left(A^k\right) = A_x A^{k-1} + A A_x A^{k-2} + A^2 A_x A^{k-3} + \cdots + A^{k-3} A_x A^2 + A^{k-2} A_x A + A^{k-1} A_x.$$

The matrices can *not* be reordered in order to find common factors in each term, since the matrices do not commute. So further algebraic operations are necessary to find an expression for the derivative. Using [1] and [9], it can be shown that

$$\begin{aligned} \tfrac{\partial}{\partial x}\exp(A) &= \operatorname{dexp}_A(A_x) \cdot \exp(A) \\ &= \exp(A) \cdot \operatorname{dexp}_{-A}(A_x) \end{aligned} \tag{3}$$

where

$$\mathrm{dexp}_A(A_x) = \sum_{m=0}^{\infty} \frac{\mathrm{ad}_A^m}{(m+1)!} A_x,$$

$$\mathrm{dexp}_{-A}(A_x) = \sum_{m=0}^{\infty} \frac{(-1)^m \mathrm{ad}_A^m}{(m+1)!} A_x.$$

dexp_A is an analytic function of ad_A,

$$\mathrm{dexp}_A = \frac{\exp(\mathrm{ad}_A) - \mathrm{I}}{\mathrm{ad}_A},$$

which is useful because it means that it does not interfere with the differentiability of functions it is applied to.

$\mathrm{ad}_A(B)$ in the expression above is the derivative of the adjoint representation $\mathrm{Ad}_P(A)$. In our case, with matrices, the ad operator corresponds to the commutator. We explore how ad works like an operator on a matrix B with respect to a matrix A:

$$\mathrm{ad}_A(B) = [A, B] = AB - BA.$$

Further, ad^2 is the operator applied twice,

$$\begin{aligned} \mathrm{ad}_A^2(B) &= \mathrm{ad}_A(\mathrm{ad}_A(B)) \\ &= [A, \mathrm{ad}_A(B)] \\ &= [A, AB - BA] \\ &= A^2B - ABA - (ABA - BA^2) \\ &= A^2B - 2ABA + BA^2. \end{aligned}$$

In general, $\mathrm{ad}_A^n(B) = \mathrm{ad}_A(\mathrm{ad}_A^{n-1}(B))$, so ad^n is the operator applied n times. ad^0 is equivalent to not applying the operator. See [5] and [7] for a discussion on using the derivative of the matrix exponential on control problems.

Now we turn to finding a zero'th order approximation to the derivative (3). We use only the first term in the Taylor series for dexp_{-A}, and get

$$\mathrm{dexp}_{-A}(A_x) \approx \frac{(-1)^0 \mathrm{ad}_A^0}{(0+1)!} A_x = A_x \tag{4}$$

which is simply the derivative of the original matrix with respect to x.

3.4 Using the matrix exponential

Now we use the matrix exponential to solve equation (2). Taking the matrix exponential of the constant matrix $-iM_{k+1}$, and setting $\Delta t = t_{k+1} - t_k$, we find that the solution of the equation at time t_{k+1} is

$$\Psi(t_{k+1}) \equiv \Psi_{k+1} = e^{-i\Delta t M_{k+1}} \cdot \Psi_k.$$

Using the matrix exponential method for solving the equation on each interval, and inserting the solution at the end of an interval into the initial condition of the problem at the next, it follows that at time $T = t_N$ the solution of the equation will be:

$$\Psi(T) = e^{-i\Delta t M_N} \cdot e^{-i\Delta t M_{N-1}} \cdots e^{-i\Delta t M_2} \cdot e^{-i\Delta t M_1} \cdot \Psi_0. \tag{5}$$

Now we will show that equation (2) preserves the magnitude of $\Psi(t)$, which means that the equation has unitary flow. First, we see that M_k is a symmetric matrix, and it follows that $-\Delta t M_k$ is also symmetric. We denote $-\Delta t M_k$ as A_k. A symmetric matrix A_k times i is skew-Hermitian, meaning that $(iA_k)^* = -iA_k$, and the matrix exponential e^{iA_k} of a skew-Hermitian matrix is, according to [3] page 25, unitary, and we denote it by U_k. According to [4] page 253, the linear transformation under a unitary matrix preserves length, that is,

$$||U_k x||_2 = ||x||_2 .$$

It follows, for our problem, that

$$||\Psi_{k+1}||_2 = \left|\left|e^{-i\Delta t M_{k+1}} \cdot \Psi_k\right|\right|_2 = ||\Psi_k||_2 .$$

So the magnitude of the solution will be the same as the magnitude of the initial value.

3.5 Measuring closeness and projecting

We want to measure how close the achieved vector $\Psi(T)$ is to the target vector Ψ_T. We do this by requiring $\Psi(T)$ to be as close as possible to the subspace spanned by Ψ_T,

$$\text{span}\{\Psi_T\} = \{\alpha\Psi_T, \alpha \in \mathbb{C}\}.$$

To measure the closeness, we need a concept of distance. Since there exists an inner product space on $\mathbb{C}^n$, there is also a distance defined by the 2-norm of the difference of vectors.

Definition 3.2. *The two-norm $||a||_2$ of a vector a is defined as $\sqrt{\langle a, a\rangle}$.*

Definition 3.3. *The inner product $\langle a, b\rangle$ on complex vectors is defined as $a^*b = \overline{a}^{\mathrm{T}} \cdot b$.*

The inner product on complex vectors has the following properties:

$$\langle a, a\rangle \geq 0$$

$$\langle a, a\rangle = 0 \text{ if and only if } a = 0$$

$$\langle a, b\rangle = \langle \overline{b}, \overline{a}\rangle = \overline{\langle b, a\rangle}$$

$$\langle Aa, b\rangle = \langle a, A^*b\rangle$$

where A^* denotes the transpose conjugate of the matrix A. The inner product is also bilinear, and satisfies the following condition:

$$\langle a, \beta b + \gamma c\rangle = \beta\langle a, b\rangle + \gamma\langle a, c\rangle.$$

It follows that

$$\begin{aligned} \langle \alpha a + \beta b, c\rangle &= \overline{\langle c, \alpha a + \beta b\rangle} \\ &= \overline{\alpha\langle c, a\rangle + \beta\langle c, b\rangle} \\ &= \overline{\alpha}\ \overline{\langle c, a\rangle} + \overline{\beta}\ \overline{\langle c, b\rangle} \\ &= \overline{\alpha}\langle a, c\rangle + \overline{\beta}\langle b, c\rangle. \end{aligned}$$

To get an expression for the distance between $\Psi(T)$ and $\text{span}\{\Psi_T\}$ we need to project the former on to the latter. For simplicity, we use the complex vectors u and v instead of $\Psi(T)$ and Ψ_T, respectively. Because we will be using start and target vectors with unit length and

because the flow of the differential equation is unitary, we require $||u||_2^2 = ||v||_2^2 = 1$. It follows that $\langle u, u\rangle = \langle v, v\rangle = 1$.

We want u to be as close as possible to the linear subspace spanned by v, $\{\alpha v, \alpha \in \mathbb{C}\}$. Therefore we define a new vector η that is the component of u orthogonal to v, that is

$$u = \alpha v + \eta.$$

Thus $||\eta||_2 = ||u - \alpha v||_2$ is the distance from u to span$\{v\}$. $||\eta||_2^2$ is the distance squared, which is easier to handle as it is equal to $\langle \eta, \eta\rangle$. Using that v and η are orthogonal and that $\langle v, v\rangle = 1$, we can show that

$$\langle v, u\rangle = \langle v, \alpha v + \eta\rangle = \alpha\langle v, v\rangle + \langle v, \eta\rangle = \alpha\langle v, v\rangle = \alpha.$$

Thus we have:

$$\begin{aligned} ||\eta||_2^2 = \langle \eta, \eta\rangle &= \langle u - \alpha v, u - \alpha v\rangle \\ &= \langle u, u\rangle - \alpha\langle u, v\rangle - \overline{\alpha}\langle v, u\rangle + \overline{\alpha}\alpha\langle v, v\rangle \\ &= 1 - \langle v, u\rangle\langle u, v\rangle - \overline{\langle v, u\rangle}\langle v, u\rangle + \overline{\langle v, u\rangle}\langle v, u\rangle \cdot 1 \\ &= 1 - \overline{\langle u, v\rangle}\langle u, v\rangle - \langle u, v\rangle\overline{\langle u, v\rangle} + \langle u, v\rangle\overline{\langle u, v\rangle} \\ &= 1 - \overline{\langle u, v\rangle}\langle u, v\rangle \\ &= 1 - |\langle u, v\rangle|^2 \end{aligned}$$

We have now shown that the distance $||\eta||^2$ is equal to $1-|\langle u, v\rangle|^2$, and we want to minimize this. The vector u that minimizes $||\eta||^2$ is obviously the same that maximizes $|\langle u, v\rangle|^2$. This is therefore a good measure of the closeness of u to the subspace spanned by v. We apply this result to our problem, and define the yield functional.

Definition 3.4. *The yield functional $\mathcal{Y}$ takes a set c of discrete values for $J(t)$ and measures how close the solution $\Psi(T)$ of equation (2) (using these discrete values) is to the subspace spanned by ψ_T. We define it as*

$$\mathcal{Y}(c) = |\langle \Psi(T), \Psi_T\rangle|^2 .$$

3.6 The cost functional

The yield functional is a fine measure of how close to the subspace spanned by Ψ_T the discrete values of $J(t)$ transform the state of the particle, but the maximum of the yield can occur at very irregular values of c_k. To avoid this when we optimize, we need to penalize erratic values of c_k's. The integral $\int_0^T (c(t))^2 dt$ (it being finite corresponding to the property of square-integrability of a continous function $c(t)$) has high value for erratic functions and low value for more nice, smooth functions. For a discrete case we can use the Riemannian sum approximation to the integral, $c^{\mathrm{T}} c\Delta t$, instead. We multiply this by a term λ which can be used to control how much erratic values should be penalized. This gives us a regularization term $\lambda\Delta t c^{\mathrm{T}} c$. We subtract it from the yield functional to give us the *cost* functional.

Definition 3.5. *The cost functional $\mathcal{I}$ takes a set c of discrete values for $J(t)$ and measures how close the solution $\Psi(T)$ of equation (2) (using these discrete values) is to the subspace spanned by ψ_T. But, because of the regularization term, it achieves lower values when the set c is erratic. We define it as*

$$\mathcal{I}(c) = |\langle \Psi(T), \Psi_T\rangle|^2 - \lambda\Delta t c^{\mathrm{T}} c.$$

We see that the regularization term simply subtracts a certain value from the yield, so that maxima at erratic values of c are removed if the coefficient λ is chosen appropriately. The cost functional will be used as the objective function in the optimization.

4 Maximizing the cost functional

In order to find the values of c_k that get $\Psi(T)$ as close to span$\{\Psi_T\}$ as possible, we need to maximize the cost functional. A local maximum of $\mathcal{I}$ is found when

$$\nabla \mathcal{I}(c) = 0,$$

so we need to compute

$$\frac{\partial}{\partial c_k}\left(|\langle \Psi(T), \Psi_T \rangle|^2 - \lambda \Delta t c^{\mathrm{T}} c\right)$$

for each value of k, set it to 0 and solve for c_k.

4.1 The derivative of $\mathcal{I}$

To differentiate $\mathcal{I}$ we need a formula for the derivative of the modulus squared. We define the function $w : \mathbb{C}^n \to \mathbb{C}$. z is a complex vector, and z_k denotes the k'th component of z. If w depends on z, we can find the needed derivative:

$$\frac{\partial}{\partial z_k}|w|^2 = \frac{\partial}{\partial z_k}(w\overline{w}) = \frac{\partial w}{\partial z_k}\overline{w} + w\frac{\partial \overline{w}}{\partial z_k}.$$

We take the conjugate of the rightmost term above:

$$\overline{w\frac{\partial \overline{w}}{\partial z_k}} = \overline{w}\overline{\frac{\partial \overline{w}}{\partial z_k}} = \overline{w}\frac{\partial w}{\partial z_k} = \frac{\partial w}{\partial z_k}\overline{w},$$

and we see that this is the same as the first term above. Along with the fact that the sum of a complex number and its conjugate is two times the real part, this gives us

$$\frac{\partial}{\partial z_k}(w\overline{w}) = \frac{\partial w}{\partial z_k}\overline{w} + \overline{\frac{\partial w}{\partial z_k}\overline{w}} = 2\mathrm{Re}\left(\frac{\partial w}{\partial z_k}\overline{w}\right).$$

We are now in a position to find the partial derivatives of $\mathcal{I}$. Using the fact that differentiation is a linear operator, that $c^{\mathrm{T}}c = c_1^2 + c_2^2 + ... + c_N^2$, that $\overline{\langle u, v\rangle} = \langle v, u\rangle$ and the results from above, we get a new expression for the derivative of $\mathcal{I}$:

$$\frac{\partial}{\partial c_k}\mathcal{I} = 2\mathrm{Re}\left(\left\langle \frac{\partial \Psi(T)}{\partial c_k}, \Psi_T \right\rangle \left\langle \Psi_T, \Psi(T) \right\rangle\right) - 2\lambda \Delta t c_k.$$

We assume the rightmost instance of $\Psi(T)$ above to be a constant vector based on the solution of equation (2) based on the values of c_k from a previous iteration. Then we can recognize $\langle \Psi_T, \Psi(T)\rangle$ as a number, and using the properties for the inner product mentioned earlier, we can move it into the leftmost inner product. The right-hand side of the remanining inner product will then be

$$\langle \Psi_T, \Psi(T)\rangle \Psi_T.$$

This can be identified as the projection of $\Psi(T)$ onto Ψ_T, and we define a *projection operator* to write it simpler.

Definition 4.1. *The projection operator P takes a vector and projects it onto Ψ_T:*

$$P\cdot = \langle \Psi_T, \cdot \rangle \Psi_T.$$

Using the projection operator we can write

$$\langle \Psi_T, \Psi(T) \rangle \Psi_T = P\Psi(T),$$

and get

$$\left\langle \frac{\partial \Psi(T)}{\partial c_k}, \Psi_T \right\rangle \left\langle \Psi_T, \Psi(T) \right\rangle = \left\langle \frac{\partial \Psi(T)}{\partial c_k}, \langle \Psi_T, \Psi(T) \rangle \Psi_T \right\rangle = \left\langle \frac{\partial \Psi(T)}{\partial c_k}, P\Psi(T) \right\rangle.$$

Thus, the derivative of the cost functional can be written even simpler as

$$\frac{\partial}{\partial c_k}\mathcal{I} = 2\mathrm{Re}\left\langle \frac{\partial \Psi(T)}{\partial c_k}, P\Psi(T) \right\rangle - 2\lambda \Delta t c_k,$$

under the assumption that $\psi(T)$ to the right comes from a previous iteration. Now we need to find the partial derivative of $\Psi(T)$ with respect to c_k for each value of k.

In the expression for $\Psi(T)$ in (5), only the factor with the matrix M_k in the exponent depends on c_k, so the rest of the factors will remain unchanged under differentiation. We have:

$$\begin{aligned} \tfrac{\partial}{\partial c_k}\Psi(T) &= \tfrac{\partial}{\partial c_k}\left(e^{-i\Delta t M_N} \cdots e^{-i\Delta t M_{k+1}} \cdot e^{-i\Delta t M_k} \cdot e^{-i\Delta t M_{k-1}} \cdots e^{-i\Delta t M_1} \cdot \Psi_0 \right) \\ &= e^{-i\Delta t M_N} \cdots e^{-i\Delta t M_{k+1}} \cdot \tfrac{\partial}{\partial c_k}\left(e^{-i\Delta t M_k} \right) \cdot e^{-i\Delta t M_{k-1}} \cdots e^{-i\Delta t M_1} \cdot \Psi_0. \end{aligned}$$

Using the derivative (3) from Section 3.3 we get:

$$\begin{aligned} \tfrac{\partial}{\partial c_k}\left(e^{-i\Delta t M_k} \right) &= \mathrm{dexp}_{-i\Delta t M_k}\left[\tfrac{\partial}{\partial c_k}\left(-i\Delta t M_k \right) \right] \cdot e^{-i\Delta t M_k} \\ &= e^{-i\Delta t M_k} \cdot \mathrm{dexp}_{i\Delta t M_k}\left(-i\Delta t M_0 \right), \end{aligned}$$

were we have used

$$M_0 = \frac{\partial M_k}{\partial c_k} = \frac{\partial}{\partial c_k}\begin{bmatrix} c_k & B \\ B & 0 \end{bmatrix} = \begin{bmatrix} 1 & 0 \\ 0 & 0 \end{bmatrix}.$$

So we get

$$\frac{\partial}{\partial c_k}\Psi(T) = e^{-i\Delta t M_N} \cdots e^{-i\Delta t M_k} \cdot \mathrm{dexp}_{i\Delta t M_k}\left(-i\Delta t M_0 \right) \cdot e^{-i\Delta t M_{k-1}} \cdots e^{-i\Delta t M_1} \cdot \Psi_0.$$

In the preceding expression for $\frac{\partial}{\partial c_k}\Psi(T)$, the terms to the right of the dexp term can be written as Ψ_{k-1}, since the repeated left-multiplication of the matrix exponentials corresponds to solving the differential equation for time t_{k-1}. Using this and inserting into the expression for the partial derivative of $\mathcal{I}$, we get

$$\frac{\partial}{\partial c_k}\mathcal{I} = 2\mathrm{Re}\left\langle e^{-i\Delta t M_N} \cdots e^{-i\Delta t M_k} \cdot \mathrm{dexp}_{i\Delta t M_k}\left(-i\Delta t M_0 \right) \cdot \Psi_{k-1}, P\Psi(T) \right\rangle - 2\lambda \Delta t c_k.$$

The matrix terms $e^{-i\Delta t M_j}$ for $j = N, \cdots, k$ in the left part of the inner product above can be moved one at a time to the right part if they are transposed and conjugated, according

to the rules for inner products on complex vectors. This is easy, as the matrices M_j are symmetric and the i factor makes the term pure imaginary. This means that

$$(e^{-i\Delta t M_j})^* = e^{i\Delta t M_j}.$$

The resulting expression for the partial derivative then becomes:

$$\frac{\partial}{\partial c_k}\mathcal{I} = 2\mathrm{Re}\Big\langle \mathrm{dexp}_{i\Delta t M_k}(-i\Delta t M_0)\cdot\Psi_{k-1}, e^{i\Delta t M_k}\cdots e^{i\Delta t M_N}\cdot P\Psi(T)\Big\rangle - 2\lambda\Delta t c_k.$$

We see now that the rightmost side of the inner product above corresponds to solving the original (discretized) differential equation from the end-point $\mathrm{P}\Psi(T)$ at time T backwards to time t_k. We call this solution χ_k, and write the discrete backwards problem in the following way:

$$\chi' = -i\begin{bmatrix} c_{k+1} & B \\ B & 0 \end{bmatrix}\chi = -iM_{k+1}\chi,\ \chi(t_{k+1}) = \chi_{k+1}. \tag{6}$$

At time t_{k-1} this problem has solution:

$$\chi(t_{k-1}) \equiv \chi_{k-1} = e^{i\Delta t M_k}\chi_k = e^{i\Delta t M_k}\cdot e^{i\Delta t M_{k+1}}\cdots e^{i\Delta t M_{N-1}}\cdot e^{i\Delta t M_N}\cdot P\Psi(T)$$

which can replace the terms in the expression for the derivative, so it becomes

$$\frac{\partial}{\partial c_k}\mathcal{I} = 2\mathrm{Re}\Big\langle \mathrm{dexp}_{i\Delta t M_k}(-i\Delta t M_0)\cdot\Psi_{k-1}, \chi_{k-1}\Big\rangle - 2\lambda\Delta t c_k. \tag{7}$$

4.2 Solving for c_k

We set $\frac{\partial}{\partial c_k}\mathcal{I} = 0$ and solve for the c_k that appears in the regularization term:

$$\begin{aligned} 0 &= 2\mathrm{Re}\Big\langle \mathrm{dexp}_{i\Delta t M_k}(-i\Delta t M_0)\cdot\Psi_{k-1}, \chi_{k-1}\Big\rangle - 2\lambda\Delta t c_k \\ c_k &= \tfrac{1}{\lambda\Delta t}\mathrm{Re}\Big\langle \mathrm{dexp}_{i\Delta t M_k}(-i\Delta t M_0)\cdot\Psi_{k-1}, \chi_{k-1}\Big\rangle \end{aligned}$$

This is as far as we can get by solving analytically for c_k, since the dexp function expands to an infinite series where each term depends on c_k. We need the low-order approximation of the dexp-function to find an explicit formula for a value $\hat{c}_k$ that approximately maximizes $\mathcal{I}$. We insert our matrix into the approximation (4) of Section 3.3, and get:

$$\mathrm{dexp}_{i\Delta t M_k}(-i\Delta t M_0) \approx -i\Delta t M_0.$$

Now we can solve explicitly for $\hat{c}_k$:

$$\begin{aligned} \hat{c}_k &= \tfrac{1}{\lambda\Delta t}\mathrm{Re}\Big\langle -i\Delta t M_0\cdot\Psi_{k-1}, \chi_{k-1}\Big\rangle \\ &= \tfrac{1}{\lambda}\mathrm{Re}\left(i\cdot\left\langle\begin{bmatrix}1&0\\0&0\end{bmatrix}\begin{bmatrix}\Psi_{k-1,1}\\\Psi_{k-1,2}\end{bmatrix}, \begin{bmatrix}\chi_{k-1,1}\\\chi_{k-1,2}\end{bmatrix}\right\rangle\right) \\ &= -\tfrac{1}{\lambda}\mathrm{Im}\Big(\overline{\Psi_{k-1,1}}\cdot\chi_{k-1,1}\Big), \end{aligned}$$

where we have used that $\mathrm{Re}(iz) = -\mathrm{Im}(z)$ and we have let $\chi_{k-1,n}$ denote the n'th component of vector χ_{k-1} and similarly for $\Psi_{k-1,n}$. This gives us an expression that can be used in the line search algorithm as an improved value of c_k, because it is closer to a zero of the gradient.

5 Applying line search to the problem

5.1 The algorithm

To set up the iterative scheme, we will use l to denote iteration number and $c^{[l]}$ and $\Psi^{[l]}$ to denote the value of c and Ψ, respectively, after iteration l. $c^{[l+1]}$ will be computed using the expression for $\hat{c}_k$ above (for each k) which is based on the previous iterate $c^{[l]}$. We will continue to use subscripts to indicate the components of the vector $c^{[l]}$. Here, we assume that input parameter Δt is chosen so that N is an integer, and that Ψ_T is known to be or chosen to be close to an achievable target. The choice of stopping conditions and the reason that we choose to save values when $\mathcal{Y}$ starts to decrease will be discussed later.

Algorithm for 1-paramter line search

input $c^{[0]}, \Psi_0, \Psi_T, T, B, \Delta t, \lambda$
$N \leftarrow T/\Delta t$
for $k = 1$ to N **do**
    calculate $M_k^{[0]}$ using $c_k^{[0]}$
    $\Psi_k \leftarrow \exp(-i\Delta t M_k^{[0]})\Psi_{k-1}$
end for
$\Psi(T) \leftarrow \Psi_N$
$l \leftarrow 0$
while (stopping conditions not met) **do**
    $l \leftarrow l + 1$
    $\chi_N \leftarrow P\Psi(T) = \langle \Psi_T, \Psi(T) \rangle \Psi_T$
    for $k = N - 1$ to 0 **do**
        $\chi_k \leftarrow \exp\left(i\Delta t M_{k+1}^{[l-1]}\right) \chi_{k+1}$
    end for
    for $k = 1$ to N **do**
        $c_k^{[l]} \leftarrow -\frac{1}{\lambda}\mathrm{Im}\left(\overline{\Psi_{k-1,1}} \cdot \chi_{k-1,1}\right)$
        calculate $M_k^{[l]}$ using $c_k^{[l]}$
        $\Psi_k \leftarrow \exp\left(-i\Delta t M_k^{[l]}\right) \Psi_{k-1}$
    end for
    $\Psi(T) \leftarrow \Psi_N$
    $\mathcal{Y}^{[l]} \leftarrow |\langle \Psi(T), \Psi_T \rangle|^2$
    $\mathcal{I}^{[l]} \leftarrow \mathcal{Y}^{[l]} - \lambda \Delta t (c^{[l]})^{\mathrm{T}} c^{[l]}$
    if $\mathcal{Y}$ has started to decrease after having increased **then**
        save $\mathcal{Y}^{[l-1]}, \mathcal{I}^{[l-1]}$ and $c^{[l-1]}$
    end if
end while

5.2 Choice of starting point

The cost functional could be very irregular and there may be many local maxima. That means that the choice of starting point for c may decide which of the many local maxima the algorithm finds. In my test runs I start at random points on the interval $[-1, 1]$ and not

explore further the effects of different starting points, and it seems that the results are very similar every time even with random starting points.

If we wanted to explore the effect of different starting points, we could to run the algorithm over and over again with evenly spaced starting points. I will sketch such an approach briefly.

Say we know that the points that maximize the cost functional are most likely within the interval $[-2, 2]$, and we want to try using points spaced at 0.4 on that interval, including the end points, (i.e. $\{-2, -1.6, -1.2, \cdots, 1.6, 2\}$) we would have 11 points for this c_k. If we have chosen $N = 200$, there will be 200 c_k values, and we would need to run the algorithm for $11^{200} \approx 1.9 \cdot 10^{208}$ points, which is impossibly expensive to compute.

But we do know that the regularization term in $\mathcal{I}$ reduces the function value at erratic controls, and also we do not want these solutions. This can be used to reduce greatly the number of starting points to test, by only choosing starting points that represent relatively smooth controls.

5.3 Choice of stopping criteria

Since our functional is rather irregular, the algorithm might give varying values for $\mathcal{I}$ and $\mathcal{Y}$ in the beginning. Therefore it is important that the algorithm is allowed to run for a while before the stopping conditions are tested, so that initial instability is corrected. We allow 10 iterations before the stopping conditions are checked. We then use the following stopping criteria:

1. **Small change in c:**
$$\left|\left|c^{[l+1]} - c^{[l]}\right|\right|_2 < tolerance.$$

2. **Small change in the cost functional $\mathcal{I}$:**
$$\mathcal{I}(c^{[l+1]}) - \mathcal{I}(c^{[l]}) < tolerance.$$

3. **Decreasing cost functional:**
$$\mathcal{I}(c^{[l+1]}) - \mathcal{I}(c^{[l]}) < 0,$$
and then we even choose $c^{[l]}$ instead of $c^{[l+1]}$ as the optimum point.

4. **Maximum iteration count reached.**

In our case it may happen that the yield $\mathcal{Y}$ starts to decrease after it has been increasing. This is of interest since the yield represents what we "really" are seeking to optimize. However, it may decrease simply because the regularization term forces the optimization to look for more smooth functions than those that have been achieved, even though they give slightly lower yield. The yield may also start to increase again after a while, and it is important that we find this. So we *don't* stop if $\mathcal{Y}$ decreases after having increased, that is

$$\mathcal{Y}(c^{[l+1]}) - \mathcal{Y}(c^{[l]}) < 0 \text{ and}$$
$$\mathcal{Y}(c^{[l]}) - \mathcal{Y}(c^{[l-1]}) > 0,$$

but instead, we save the values of $c^{[l]}$, $\mathcal{Y}(c^{[l]})$ and $\mathcal{I}(c^{[l]})$. When the iteration stops due to one of the above stopping conditions, we can compare these saved values to the final values. If $\mathcal{Y}$

at the first point is larger than at the last point and if the difference in $\mathcal{I}$ between the two points is very little (meaning that the controls at the first point are almost as smooth as at the second), we might as well use the first value as the optimum point. If the difference in $\mathcal{I}$ is large, we must use the final values instead.

5.4 Choice of start and target Ψ

The physical properties of the problem are beside the point of this paper, so the choice of starting state Ψ_0 is arbitrary for our purposes. But the target Ψ_T requires careful consideration.

First of all, the fact that the differential equation (1) has unitary flow means that the solution of the equation will have the same magnitude as the initial condition regardless of the controls, i.e. $||\Psi_0||_2 = ||\Psi(T)||_2$. So we must choose Ψ_T so that it satisfies $||\Psi_T||_2 = ||\Psi(0)||_2$, and since we will only be using starting vectors of unit length, we require $||\Psi_T||_2 = 1$.

Secondly, Ψ_T must be possible to reach as a solution of the differential equation at time T. To find a target state that definitely is reachable, we can discretize a known function, for example $\cos(t)$, at N points from 0 to T, and set c_k equal to its function values at each point in time. We insert these into the differential equation (2) and solve for $\Psi(T)$. We use this value as Ψ_T when we run the optimization. In this way, we know that there exist values of c_k that allow Ψ_T to be reached. However, these values of c will not necessarily be reached exactly through the optimization, because of the regularization term $\lambda c^{\mathrm{T}} c$ in $\mathcal{I}$. This is because λ in that term slightly changes the space of values that c can achieve.

6 Results

The aim of this section is to explore how the mathematical concepts and optimization method introduced to the original problem affect the iterative solution of the problem. There are two input parameters that are entirely unrelated to the original physical problem, namely λ and Δt. The choice of these two parameters will decide on how close the solution $\Psi(T)$ can get to Ψ_T, how irregular the calculated values of c will be and also how many iterations are required. So the effects of varying λ and Δt will be explored.

The parameters B, T, Ψ_0 and Ψ_T are purely physical, and it is not within the scope of this paper to consider these. However, what should be considered is the effect that the changing of these physical parameters has on how we should choose the mathematical parameters. In order to do this, I will find a good choice for λ and Δt that works for certain values of the others. Next, I will alter B and T one at a time to see the effect on the solution and also on the ability for the algorithm to find a solution in the first place.

6.1 An example run

First we show a typical example run of the algorithm. Input parameters are:

$$\begin{aligned} \Psi_0 &= \begin{bmatrix} 1 \\ 0 \end{bmatrix} \\ T &= \pi \\ B &= 1 \\ \Delta t &= 0.01 \\ \lambda &= 0.2 \\ tolerance &= 10^{-6} \\ maxIter &= 100 \end{aligned}$$

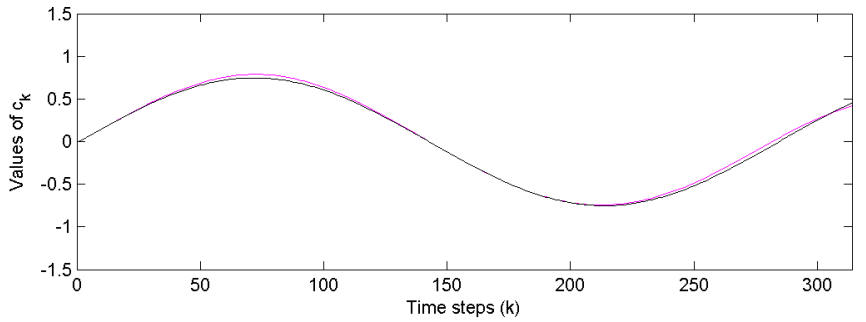
Using these, a few parameters must be calculated. We re-calculate Δt so that Δt times N is exactly equal to T, so we get a generated interval size $\Delta t_{gen} = 0.010005$. We find Ψ_T by applying $J(t) = \cos(t)$ on the interval $[0, T]$ divided into N points and solving for $\Psi(T)$, so we choose

$$\Psi_T = \Psi(T) = \begin{bmatrix} -0.7869 \\ -0.5687 - 0.2396i \end{bmatrix}$$

which means that Ψ_T is a point that we know is reachable by a piecewise constant control function. We run the algorithm, and present the results in a table and two graphs:

Table 1: An example run

λ	Δt_{orig}	Δt_{gen}	N	I_{final}	Y_{final}	I_{ymax}	Y_{ymax}	Iter	Stop
0.1	0.01	0.010005	314	0.88512	0.96669	0.88492	0.96855	19	I-small



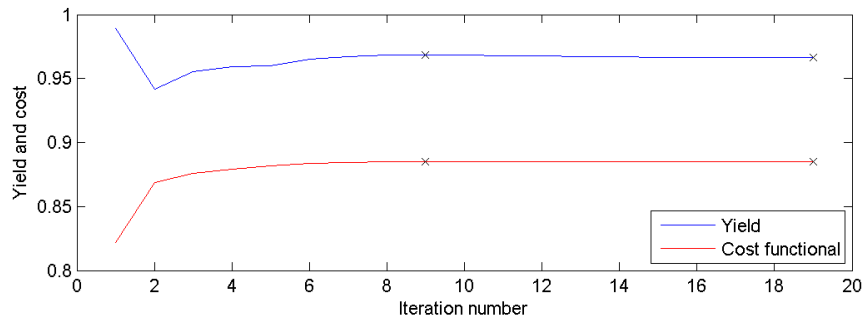


Table 1 contains the input parameters, Δt_{gen} generated to reach the target time exactly and the number of discrete points. Also, the resulting values of $\mathcal{Y}$ and $\mathcal{I}$ after optimization are shown, both at the maximum found for the yield and for the cost. At the end, the number of iterations used and the reason that the iteration stopped are shown. The reasons are I-desc (descending $\mathcal{I}$), I-neg (negative $\mathcal{I}$[1]), I-small (small change in $\mathcal{I}$), c-small (small change in c) or max (maximum iterations). The upper graph shows the values of c_k for both maxima, the one for $\mathcal{I}$ in black while the one for $\mathcal{Y}$ in magenta, while at the lower graph we see the development of the cost and yield functionals as the iteration number increases. Both maximuma are marked with an x for each functional at both points, the rightmost being of course the maximum for $\mathcal{I}$, since this is when we stop the iterations.

We see that the yield has a maximum already after half of the iterations needed to meet the stopping conditions, and that the difference of the cost functional at the two different maximum points is very small. This means that the controls that give the maximum for the yield are probably regular enough to be used as a final solution. This is confirmed by looking at how close the controls in the upper graphs are, and that they both seem quite regular.

6.2 Varying Δt

We now run the algorithm for 6 different values of Δt, while we fix λ at 0.1 and keep all the other parameters as in the test run. The results are used to find an optimal spacing for the algorithm, and are summarized in Table 2 and shown as a series of graphs. Here, # denotes run number.

Table 2: Results from varying Δt when $\lambda = 0.1$

#	Δt_{orig}	Δt_{gen}	N	I_{final}	Y_{final}	I_{ymax}	Y_{ymax}	iter	stop
1	0.5	0.5236	6	-0.45015	0.90519	-1.008	0.98968	11	I-neg
2	0.1	0.10134	31	0.87984	0.96321	0.87866	0.96954	20	I-small
3	0.05	0.050671	62	0.88299	0.96527	0.88283	0.9675	19	I-small
4	0.02	0.02001	157	0.88463	0.96637	0.88456	0.96776	18	I-small
5	0.01	0.010005	314	0.88512	0.96669	0.88494	0.96846	19	I-small
6	0.005	0.0050025	628	0.88536	0.96682	0.8851	0.9689	17	I-small

We see that using a large value for Δt such as 0.5 does not allow the function to be smooth enough to allow the cost functional to achieve high values. This is simply because there are not enough points. Also, the regularization term depends on Δt and is large when Δt is large, so the cost may be much smaller than the yield. Also, the yield and cost vary a lot, indictating that the iterations do not converge.

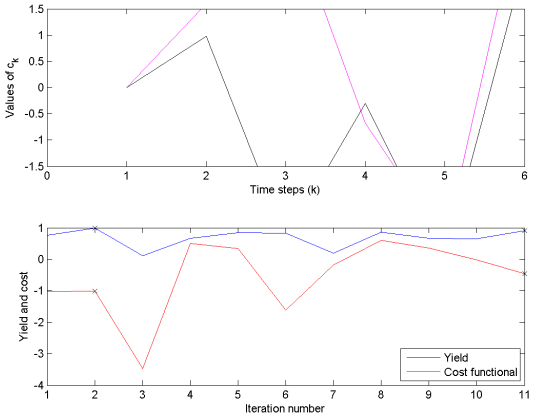
As soon as there are enough points to allow c to be a relatively smooth discrete function, such as with $\Delta t_{orig} = 0.1$, the optimization results in a gradually increasing cost functional and a very smooth, trigonometric-looking control function. But we see from the table that the values of the cost functional are quite similar at the maximum for the cost and at the maximum for the yield. We also see that the maximum yield and cost increase a little as the constant intervals become finer and finer, also without increasing the number of iterations

[1]This was added because divergence can give a negative cost functional, even though it might not be decreasing. Remember that the stopping criteria are not checked before iteration 10 is reached, so this can occur.

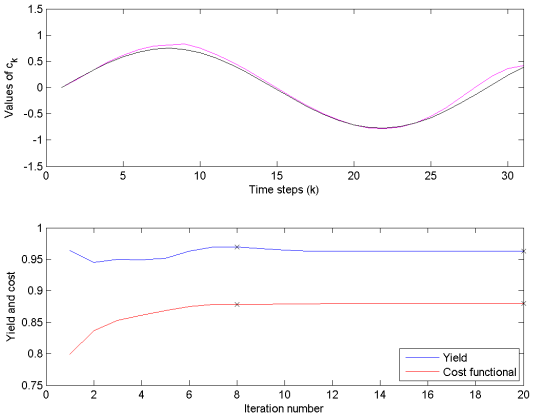
significantly. We conclude that $\Delta t_{orig} = 0.01 \Rightarrow \Delta t_{gen} = 0.010005$, corresponding to using 314 discrete points for this interval, is a good choice because it gives high yield and the yield does not seem to improve much if we had used two times as many points instead.

Graphs of varying Δt

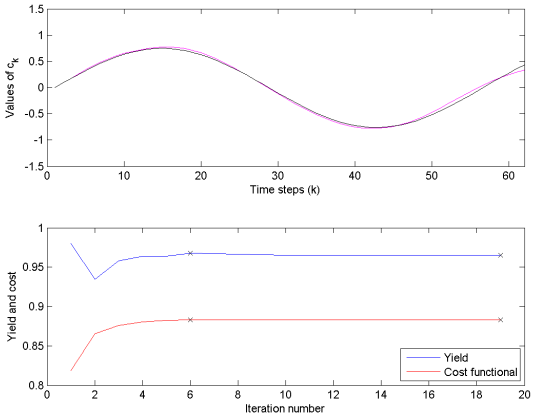
$\#1, \Delta t = 0.5$:



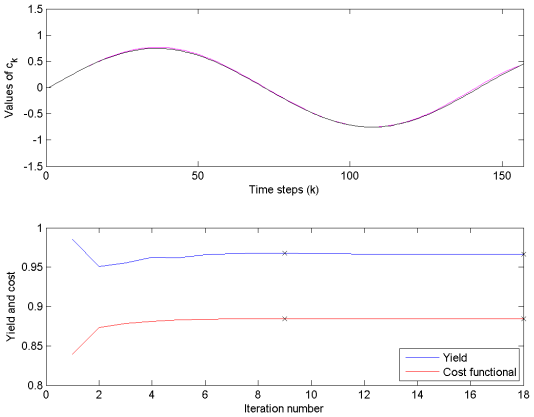
$\#2, \Delta t = 0.1$:



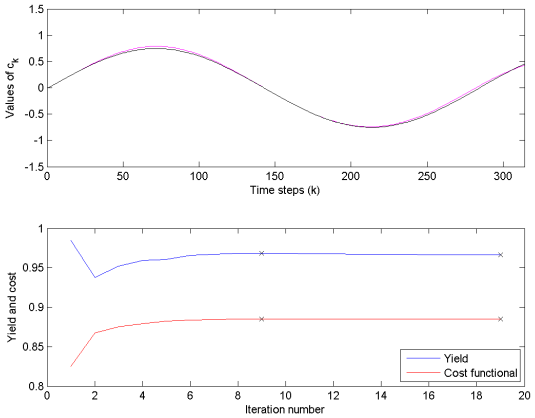
$\#3, \Delta t = 0.05$:



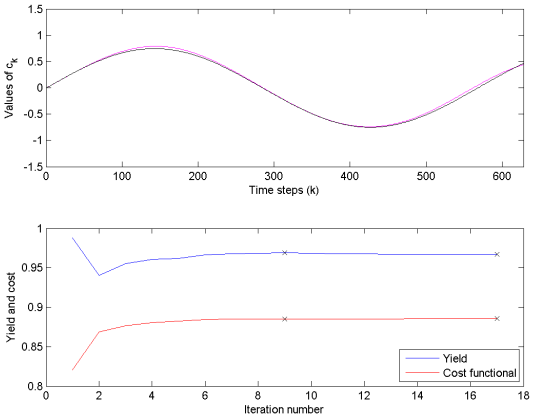
$\#4, \Delta t = 0.02$:



$\#5, \Delta t = 0.01$:



$\#6, \Delta t = 0.05$:



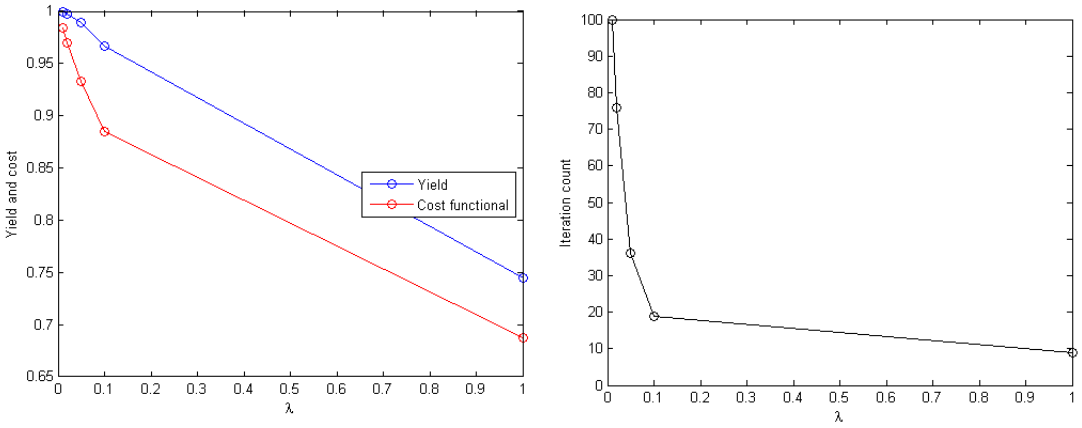
6.3 Varying λ

We now use the optimal value for Δt that we just found, and see what happens when we vary the coefficient of the regularization term instead. The other parameters are kept the same as in the test run. Table 3 shows the results, and the graphs of the solutions follow below. The results for the varying of λ can also be summarized in a graph of λ versus $\mathcal{I}$ and $\mathcal{Y}$, and λ versus iteration count[2]. We see that run $\#7$ misses values for the maximum of the yield, and this is because it turned out to be the same as the maximum for the cost. The result from run $\#12$ is excluded in the graph because the cost was very negative, and it would be too difficult to interpret the results of the other runs with the y-axis that would be needed to display it.

[2]These two graphs are not based on the same runs as the results in the table, but they are done with the exact same parameters (the only difference being the random starting point for c).

Table 3: Results from varying λ when $\Delta t = 0.010005$ and $N = 314$

#	λ	I_{final}	Y_{final}	I_{ymax}	Y_{ymax}	iter	stop
7	1	0.68716	0.74474	.	.	9	c-small
8	0.1	0.88512	0.96668	0.88494	0.96853	18	I-small
9	0.05	0.93303	0.98888	0.93227	0.99062	35	I-small
10	0.02	0.9703	0.99787	0.96887	0.99852	76	I-small
11	0.01	0.98424	0.99928	0.98135	0.99973	100	max
12	0.005	-89.045	0.99714	-45.671	0.99682	11	I-neg



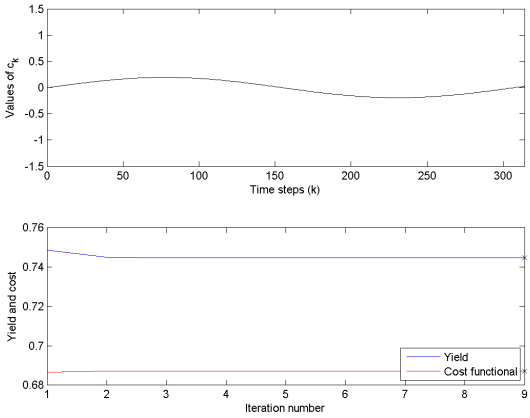
The main point of the regularization term is to exclude highly erratic solutions. As long as the solution looks smooth enough when plotted, we can probably use it. We see that run #7 where λ is high gives poor yield and very flat control functions, while decreasing the λ gives better and better yield and cost up to a certain point. Naturally, the difference between yield and cost is smaller as λ is decreased. But at the same time, the number of iterations before a stopping condition is met increases rapidly (the graph for run #11 was not stopped at 100 iterations, in order to demonstrate that the result improves very slowly and very little after this). Also, the difference in the cost between the maximum for cost and the maximum for yield increases with λ. We see this in runs #10 and #11 where the magenta-coloured lines are more different from the black ones than they are in the previous runs. Here, a trade-off must be made. If the magenta curve in those runs is considered smooth enough, it can be used, and it gives a higher yield than the final iteration. However, in order to *know* that this is a maximum for the yield, we must run the algorithm until the change in the cost functional is so small that we believe it to be unlikely that further iterations will give a higher yield. Since the number of iterations between the maximum of the yield and of the cost is so large, it indicates that perhaps a higher tolerance should have been used.

From these results, it seems that $\lambda = 0.02$ can be a good choice, because at run #10 the magenta curve definitely seems smooth enough, and it is reached in less than 30 iterations.

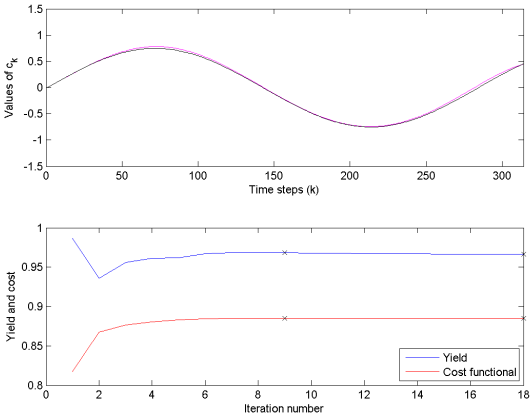
When λ is set to 0.005 in run #12, we see that we loose control and get highly oscillating controls, and even though the yield is high, the cost functional is very negative and decreases rapidly. This is because the regularization term is too small to have the desired effect of removing maxima of the cost functional at erratic values of c.

Graphs of varying λ

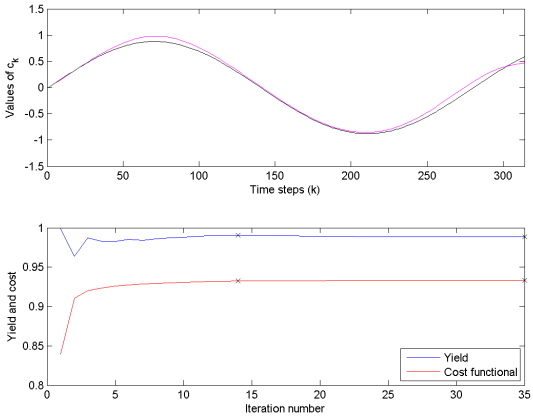
$\#7, \lambda = 1$:



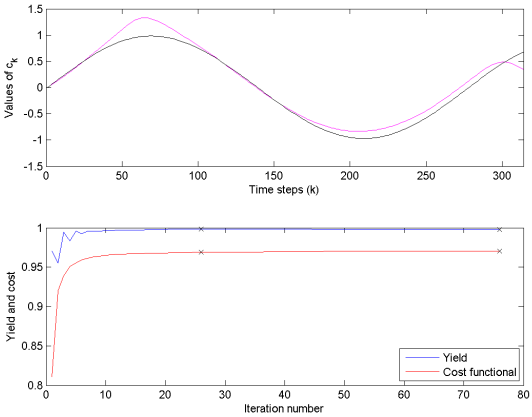
$\#8, \lambda = 0.1$:



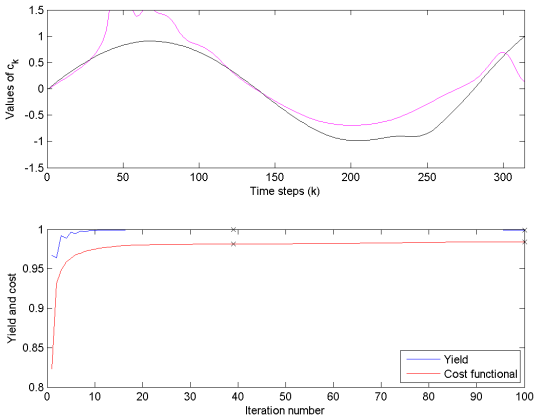
$\#9, \lambda = 0.05$:



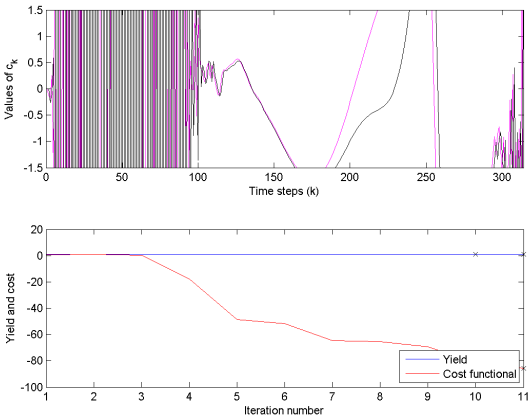
$\#10, \lambda = 0.02$:



$\#11, \lambda = 0.01$:



$\#12, \lambda = 0.005$:



6.4 Varying B and T

Now we look at the effects of varying the physical quantity B and the end-point of the time interval, while keeping the other parameters as in the example run. Also, we keep $\lambda = 0.02$ and $\Delta t = 0.01$. See the results in Table 4 and the graphs following.

Table 4: Results from varying B and T

#	T	B	I_{final}	Y_{final}	I_{ymax}	Y_{ymax}	iter	stop
13	1.5708	0.5	0.7791	0.9346	0.7118	0.9692	90	I-small
14	1.5708	1	0.9883	0.9983	0.9875	0.9993	50	I-small
15	1.5708	2	0.9953	0.9973	0.9942	0.9979	33	I-small
16	3.1416	0.5	0.9959	0.9966	0.9956	0.9966	75	I-small
17	3.1416	1	0.9703	0.9979	0.9679	0.9987	75	I-small
18	3.1416	2	0.9925	0.9986	0.9771	1.0000	68	I-small
19	6.2832	0.5	0.9995	1.0000	0.9982	1.0000	100	max
20	6.2832	1	0.9775	0.9986	0.9727	0.9993	100	max
21	6.2832	2	0.9845	0.9985	0.9827	0.9994	100	max

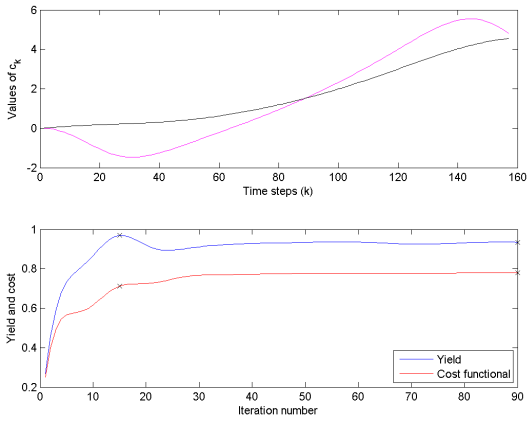
Because the width of the time-interval changes while Δt is kept constant, the number of points increases with T.

We see from the graphs that for $B = 0.5$, the control functions seem to be very flat, indicating that for lower B we should have smaller λ. Also, we see that that high B seems to greatly increase the difference in shape of the discrete control function at the points where $\mathcal{I}$ and $\mathcal{Y}$ are maximized, respectively.

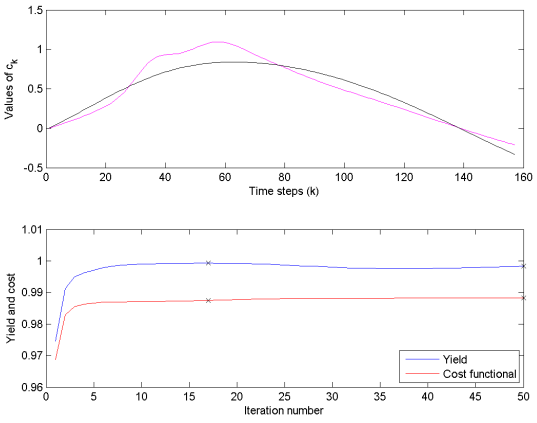
Another observation is that the number of oscillations in the optimal control function seems to increase as the time interval increases, and as B increases. Also, when $T = 2\pi$, a maximum of the cost functional is not found because the method is too slow to converge and it runs out of allowed iterations. This suggests that we have used too many discrete points and should reduce Δt for this choice of interval.

Graphs of varying B and T

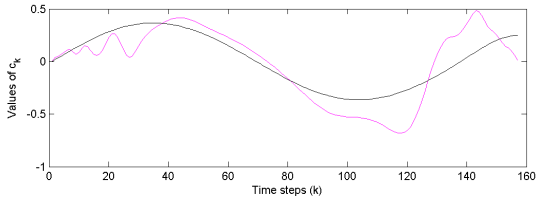
$\#13, T = \pi/2, B = 0.5$:

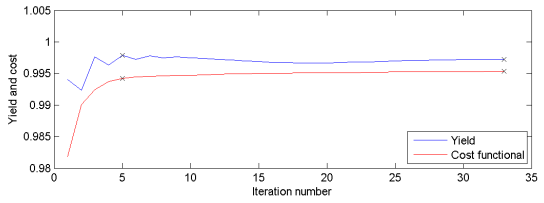


$\#14, T = \pi/2, B = 1$:

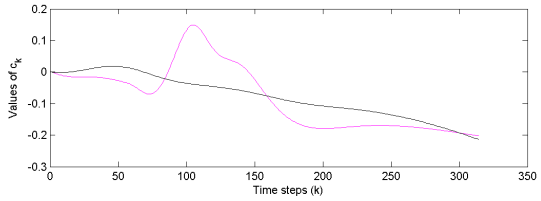


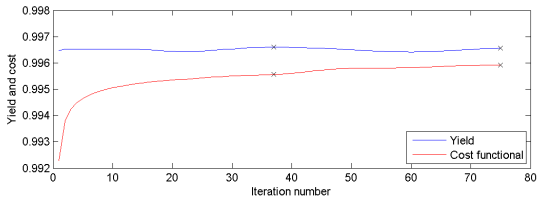
$\#15, T = \pi/2, B = 2$:



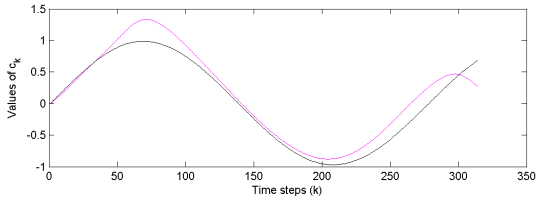


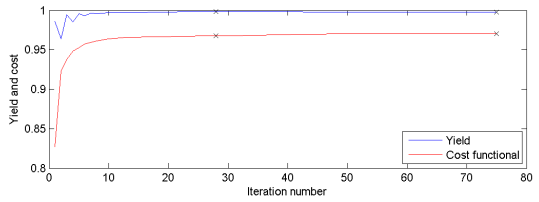
$\#16, T = \pi, B = 0.5$:



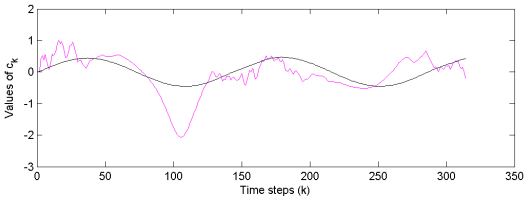


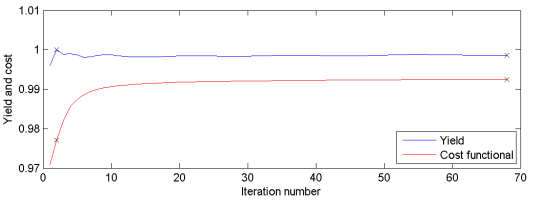
$\#17, T = \pi, B = 1$:



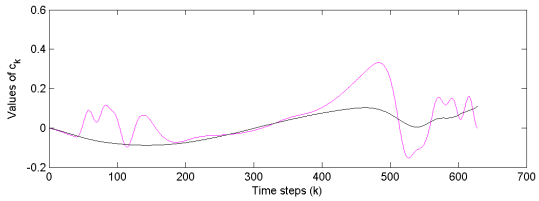


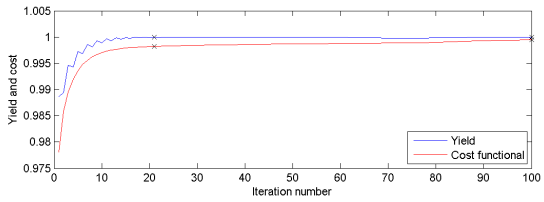
$\#18, T = \pi, B = 2$:



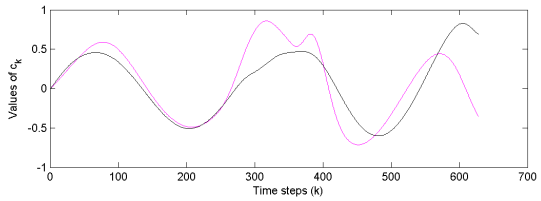


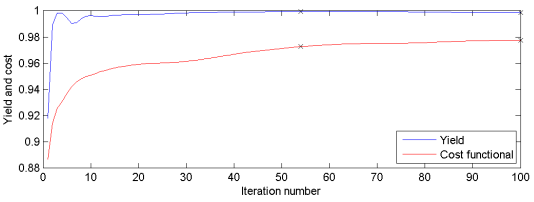
$\#19, T = 2\pi, B = 0.5$:



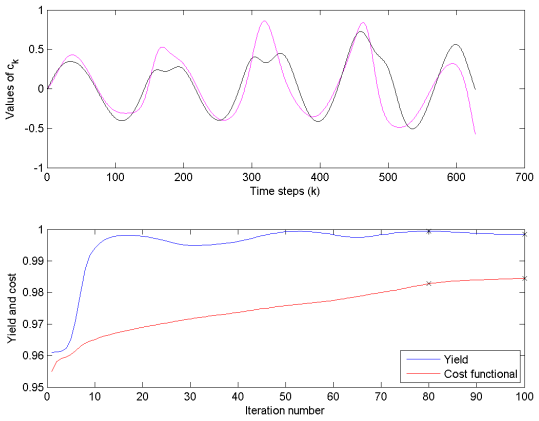


$\#20, T = 2\pi, B = 1$:





#21, $T = 2\pi, B = 2$:



7 Conclusion

We have seen that applying optimization to quantum control problems is not straight-forward, and that great care must be taken in choosing methods and parameters to get the desired result. We have seen that

- the nature of the optimization problem is important in choosing the right algorithm and stopping criteria. Knowledge of the mathematical structure of the problem helps in choosing a method that converges more quickly.
- different choices of starting points do not seem to change the result of the optimization very much.
- the algorithm may introduce instabilities not present in the original problem, for example we saw that the λ changed the space of allowed controls so that the target state made with the discretized values of $\cos(t)$ was not possible to achieve exactly.
- optimal choice of mathematical parameters depends on the physical parameters of the exact problem.

It is perhaps not possible to find the right parameters without doing several trial runs that do not give the desired result, and fine-tune the parameters till the desired result is achieved.

References

[1] S. P. Nørsett A. Iserles, H. Munthe-Kaas and A. Zanna. Lie-group methods. *Acta Numerica*, 9:248–249, 2000.

[2] W. E. Boyce and R. DiPrima. *Elementary Differential Equations and Boundary Value Problems.* Wiley, 8th edition, 2004.

[3] N. J. Higham. *Functions of Matrices : theory and computation.* Philadelphia, Pa. : Society for Industrial and Applied Mathematics, 2008.

[4] Franz E. Hohn. *Elementary Matrix Algebra.* New York : Macmillan, 2nd edition edition, 1964.

[5] L. Sælen I. Degani, A. Z. Munthe-Kaas and R. Nepstad. Quantum control with piecewise constant control functions. To appear, SIAM J. Sci. Comput., 2009.

[6] V. F. Krotov. *Global methods in optimal control theory.* Martcel Dekker, Inc., New York, 1996.

[7] A. Z. Munthe-Kaas. A note on the exact dexp expansion for the control of quantum spin systems. Personal note, 2008.

[8] J. Nocedal and S. J. Wright. *Numerical Optimization.* Springer Series in Operations Research. Springer Science+Business Media, LLC, 2nd edition, 2006.

[9] V. S. Varadarajan. *Lie Groups, Lie Algebras and Their Representations.* Graduate Texts in Mathematics. Springer-Verlag., 1984.

[10] W. Zhu and H. Rabitz. A rapid monotonically convergent iteration algorithm for quantum optimal control over the expectation value of a positive definite operator. *J. Chem. Phys.*, (109):385–391, 1998.