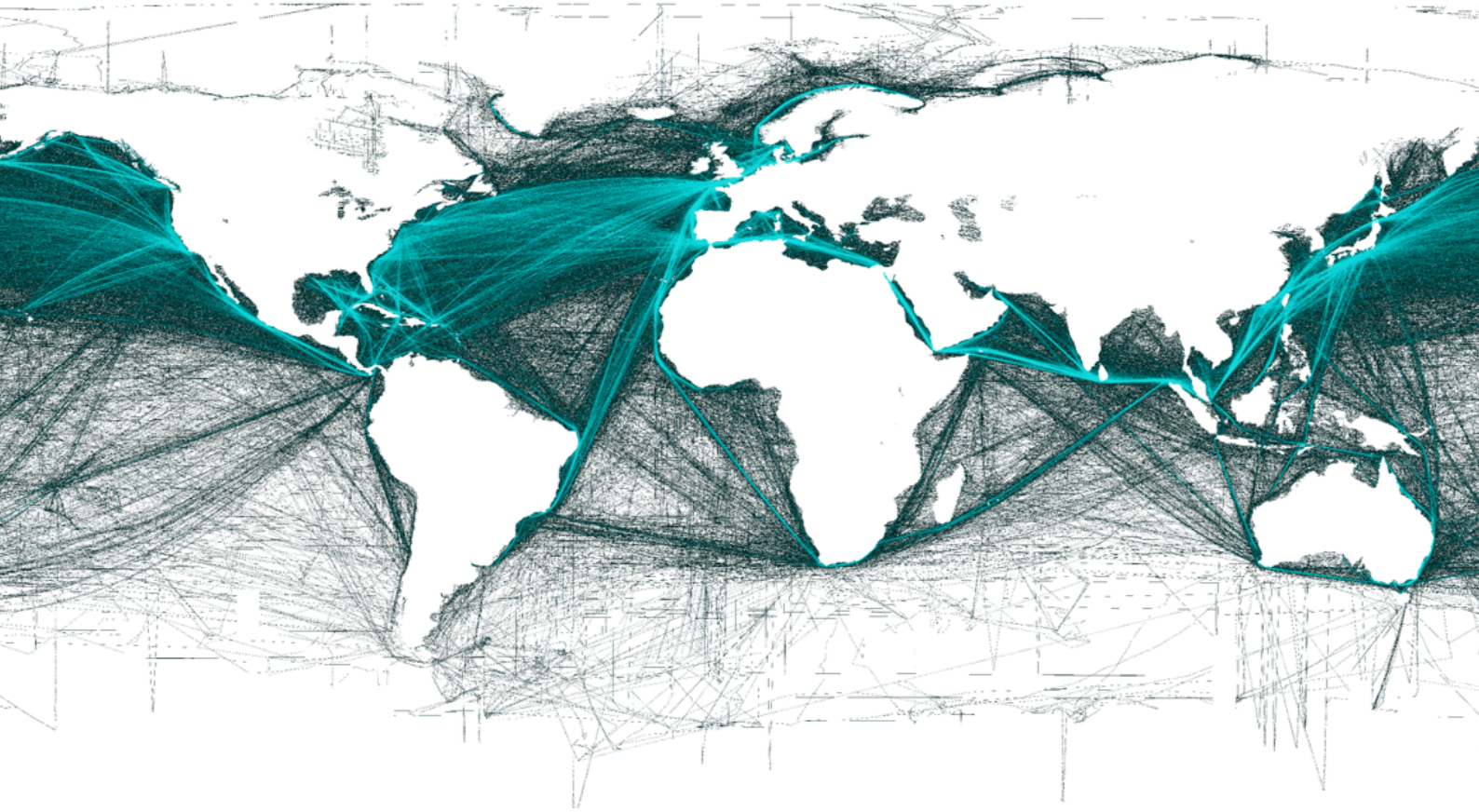
A Non-Convex Path Optimization Analysis for Reduced Ship Travel Time

An engineering approach



Kirsten Odendaal

Supervisors:
Dr.ir. Matthijs Langelaar

Delft University of Technology

List of Symbols

Latin

f_{obj}	Objective function	[]

Other

H	Optimization function Hessian	[]
x	Continuous way point design variables	[km]
$\vec{V}_{curr}$	Current Vector magnitude	[km/hr]
$\vec{V}_{x,curr}$	Current Vector in spacial x-direction	[km/hr]
$\vec{V}_{y,curr}$	Current Vector in spacial y-direction	[km/hr]
dt	Incremental time scale	[hr]
dx	Incremental length scale	[km]
E	Energy function (Simulated Annealing)	[hr]
F	Effective current direction magnitudes	[km/hr]
f	Objective function	[]
g	function inequality constraint	[]
H	modified heavy-side step function	[hr]
L	Effective spacial magnitude	[km]
n	Mesh refinement factor	[]
P	Probability of accepting worse solution (Simulated Annealing)	[]
SZX	Maximum domain size in x-direction	[km]
SZY	Maximum domain size in y-direction	[km]
T_{frac}	Decreasing temperature fraction (Simulated Annealing)	[]
T_{travel}	Total time travelled	[hr]
V_{ship}	Ship Velocity	[km/hr]
X_{grid}	Grid sizing in x-direction	[km]
Y_{grid}	Grid sizing in y-direction	[km]

Contents

1 Introduction

Optimizing a ship's voyage is one of the main objectives and challenges of any shipping company to be cost competitive. It is driven by the need to operate a ship as cost efficient, energy efficient and safe as possible during every voyage in its lifetime. As such, voyage planning is supported by weather routing systems on many vessels nowadays. Therefore, this paper will investigate multiple optimization algorithms, and establish a best tool when information about the problem is not well known and/or multiple variables must be considered. First, a brief description of the background and motivation behind path optimization will first be addressed. Next, the fundamental problem formulations will be established and explained in detail. This will be followed by a description of the black-box ship routing model as well as the key considerations and assumptions implemented. The problem will then undergo an initial investigation where boundedness, convexity, numerical noise, and sensitivity will be addressed. This will lead into a simplified problem investigation where three optimization algorithms will further be analyzed: Quasi-Newton BFGS, Particle Swarm, and Simulated Annealing (self-scripted). Next, a more advanced routing problem will be implemented with multiple variables and obstructions. Finally, using all previous results and observations, conclusions as to which optimization algorithm performs the best and further study recommendations will be considered.

1.1 Background

Path optimization is not uncommon. Throughout history man has tried to travel the quickest, or the most efficient from one point to another. The most efficient as otherwise valuable resources would have been used. Today this accounts for being able to sell or offer something at a lower cost than competitors. Others have taken pride in being quicker than anyone to a designated point. In maritime applications both forms of objectives express themselves. For example, when the America's have been discovered, many have took up the challenge to call themselves the quickest to cross the Atlantic. A great example is the schooner Atlantic which set the record in 1905 and remained unbeaten until 1997 by the yacht Nicorette. A good example for efficiency is the overall trend in slow steaming in transporting goods. This trend was observed in 2007 when the oil prices were rapidly increasing. Because of this the Emma Mearsk which was the largest container ship at that time had to sail at much lower speeds than what she was designed for. As such, history has constantly proven the demand and desire for efficient and fast sailing.

1.2 Motivation

The motivation for this project is to become acquainted with path optimization for ships. Path optimization for ships is increasingly used both in sports, as in industrial applications. This is a problem that can be simplified to a great extend, but in reality can be very complex once weather and current models are implemented. Further complexity is added when barrier methods are added by means of islands. In summary, this investigation is thus a great starting point to observe the effects and become familiarised with optimization and the various methodologies associated.

2 Problem Formulation

2.1 Optimization Problem

Initially the problem will be optimized for the least amount of travel time. This could also be used to further minimize power based on an equality constraint for the travel time. The main challenge of this problem is that the path is defined by means of so-called Way points. The position of these Way points can vary continuously in both the x as in y directions. Therefore, the number of design variables scales with $2n$, with n being the number of Way points. The optimization problem can thus be generally formatted in the following way,

$$\begin{aligned} \min_{\mathbf{x}} \quad & f(\mathbf{x}) \\ \text{subject to} \quad & g_i(x) \leq b_i,\ i = 1, \ldots, m \\ & \underline{x} \leq \mathbf{x} \leq \bar{x} \\ & \mathbf{x} = (x_j, y_j)\ x, y \in \mathbb{R},\ j = 1, \ldots, n \end{aligned}$$

In the equation above, the general mathematical statement of the problem is shown. The form used is a standard negative null form with just one inequality constraint and no equality constraints. This statement will become more clear when reading further chapters, however a brief explanation of each component can be found below,

- **Objective function:** The objective function is define as $f(\mathbf{x})$. This function should be minimized with respect to the design variable **x**. This function uses numerical methods within MATLAB to determine the time required to travel a possible sea route. It contains a variety of steps and equations; thus an analytical expression for this function is impossible to determine.

- **Design variables:** The design variable,**x**, consists of two variables: x_j and y_j ; these are coordinate WayPoint locations. The coordinates all are real and continuous values within an established domain. The amount of coordinates is denoted by j and runs from 1 until n, where n is the amount of user established WayPoints. These WayPoints, as described in the next chapter, are user established values which define a path from a start to end location.

- **Upper and lower bounds:** The boundary limits are defined by both the upper,$\bar{x}$, and lower,$\underline{x}$, limits of the domain. In the case of the study, the domain ranges from $0\,km < x_i < 500\,km$ in the x-domain, and $0\,km < y_i < 300\,km$ in the y-domain. These domain boundaries are considered restrictions for the WayPoints as they cannot leave this area.

- **Inequality constraint:** While a constraint is applied in the mathematical description, the model actually converts a constrained problem into that of an unconstrained optimization problem using a Barrier implementation approach. This methodology, allows for the addition path obstructions by forcing the objective function to increase as consequence.

$$\tilde{f} = f - \frac{1}{r} ln(-g) \tag{2.1}$$

  Where g, represents the inequality constraint, and r, is a steepness tuning parameter. While mathematically, the barrier description can be defined as above, numerically the implementation of the barrier function deviates slightly, this procedure will be explained in more detail in the coming chapters.

2.2 Modelling Aspects and Assumptions

The problem is formulated in a rectangular grid design space with spatial dimensions x and y. On this grid a random current is projected with magnitude and direction. The model ultimately, creates a series of Way points and interpolates a line on an associated velocity vector field to determine the time of travel. An additional study into the effects of route obstructions will be analyzed along with routine responses. Ultimately, this model is implemented as a tool to evaluate multiple optimization algorithms at the expense of computational time. As such, model simplifications and assumptions were applied to ensure complexity and computational demand remained reasonable. The main assumptions made in the model are;

- **Fixed domain:** The size of the domain will be fixed and all paths will lie within the set boundary limits as defined in the problem formulation. While the domain remains fixed, the grid size will allowing for mesh refinement.
- **Path points:** Current vectors experienced along the path will be linearly interpolated.
- **Path line:** The path line will be plotted as a Piece-wise Cubic Hermite Interpolating Polynomial (PCHIP), to ensure smooth transitions. Actual turning radii and ship maneuverability formulations will not be considered.
- **Ship velocity:** Vessel speed will stay constant without regard to engine effects. This study will be consider a vessel travelling at a constant ship speed of $V_{ship} = 40$ km/hrs (~ 20 knots).
- **Obstruction objects:** The shapes will consist of basic geometrical shapes (Squares) to represent obstructions such as islands and/or other stationary objects. These obstructions represent constraints which are converted to an unconstrained problem using a barrier methodology.

2.2.1 Domain and Current Generation

The domain is generated by establishing a predetermined size in both the spatial x and y coordinates. This domain is then gridded, where a series of smooth velocity vectors can be applied to each coordinate direction (x and y) respectively. This method of creating velocity vectors is done individually through use of a random generating function, which is then interpolated to the gird using a Fourier transform methodology to ensure the velocity vectors are smooth throughout the entire field.

To ensure consistency throughout the study, a random 'seed' was established. This is a number used to initialize a pseudo-random number generating function. Thus, with the implementation of such a parameter, the randomness of functions ultimately disappear as the sequencing is then fixed for each established 'seed'. It should be noted that this seeding does indeed lead to consistent results, however certain optimization routines require a degree of randomness to successfully operate. Therefore, the seeding is removed for the optimization routines and only applied to generate a consistent domain. For the sake of this study a seed of 20 was selected. This was ultimately an arbitrary selection, however the vector field solution provided an interesting division of flows.

Once a domain and current vector field is established, an appropriate visualization of the domain data is required to make the ultimate path travelled easier to understand. This is done through what is known as a favorability criterion. Essentially, this parameter relates cell vector orientation to either a positive or a negative depending on whether the current aids ship travel or hinders it through linear interpolation.

$$F = \frac{(SZX - X_{grid}) \cdot V_{x,curr} + (SZY - Y_{grid}) \cdot V_{y,curr}}{L} \tag{2.2}$$

where,

$$L = \sqrt{(SZX - X_{grid})^2 + (SZY - Y_{grid})^2} \tag{2.3}$$

In this case, $V_{x,curr}$ and $V_{y,curr}$ is the current velocity in the associated directions, SZX and SZY is the maximum domain lengths in both the spacial directions, X_{grid} and Y_{grid} the full matrix domain, and L is a normalization of these effective spacial coordinates. Thus, this leaves a matrix which expresses the magnitudes of the current as either a positive favorable, or a negative non-favorable value. By applying this favorability field as a color map over the existing velocity domain, a clear visualization of the currents can be determined as seen in figure (1).

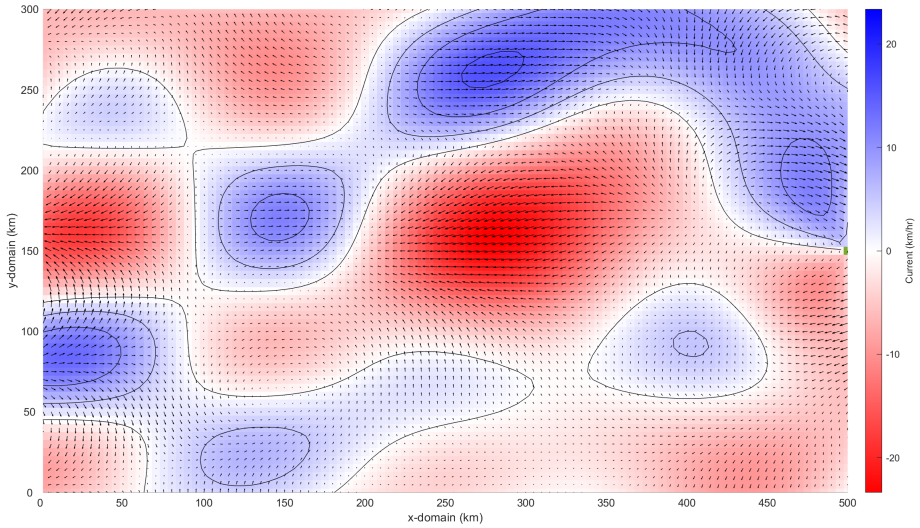


Figure 1: Full domain plot with velocity current favorability (n = 2)

For this cases study, it can be interpreted that the maximum and minimum currents are approximately 20 km/hr (5.56 m/s) respectively. Therefore, without an associated vector field (i.e. no ship resistance), the time of travel from start to finish (linearly) would take 12.5 hours, if the vessel is travelling at the known constant speed of 40 km/hr. While the grid appears quite course, the script has the capability of applying a grid refinement factor, n, to allow for a thorough sensitivity analysis while ensuring the parameters are scaled appropriately.

2.2.2 WayPoints and Interpolation

The “Path” through this domain is subdivided into multiple WayPoints. The number of WayPoints can be independently selected and placed as continuous variables within the model. From each of the points, a corresponding best fit line is created using a special polynomial interpolating function to ensure each WayPoint is intersected smoothly. This line can be further subdivided into multiple individual segments, dx, which are then linearly interpolated onto the velocity vector domain. This allows for each line component to be composed of an associated velocity component. Which is used to determine the incremental time, dt, for each segment. These segments are added to determine

the total time of travel. The model thus implements a numerical method instead of an analytical formulation to determine total time. The overall objective outcome can be described as,

$$f = T_{travel} = \sum_{n=1}^{N} dt = \sum_{n=1}^{N} \frac{dx}{(V_{ship} + V_{curr})} \tag{2.4}$$

In the above formulation, both dx and V_{curr}, are dependent on the location of the WayPoints and the interpolation techniques applied. As such, the controlling continuous WayPoints will be the design variables of the objective function.

2.2.3 Island Generation

while the above model is capable of providing an interesting and variable situation for optimization routines, an additional component was implemented to truly evaluate the effectiveness of various methodologies. Obstructions, or path hindrances were therefore incorporated. These consist of randomly generated ”islands” placed within the domain. While, the size of the islands are randomly generated, the locations are positioned using a Latin Hyper-cube Sampling method, LHS.

This is a statistical method for generating a near-random sample of parameter values from a multidimensional distribution. This procedure, while generally used only for model value sampling, is effective due to the nicely distributed location placements within the working domain. Due to the random seeding mentioned previously, the consistency of the islands can also be ensured. Figure (2), highlights the domain with 5 obstructions in the working field.

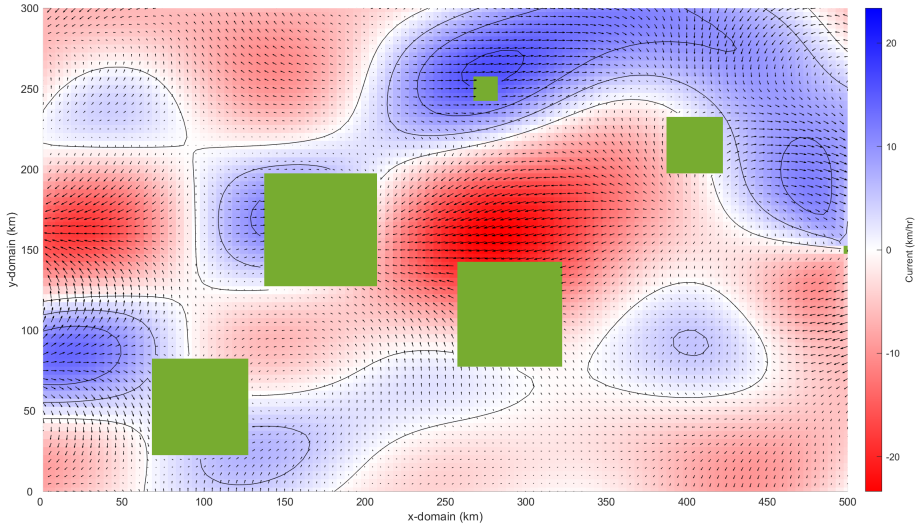


Figure 2: Full domain plot with LHS generated islands (n=2)

Modified Barrier Method: Functionally, these regions in the vector field can be represented as individual constraints applied to each design variable. However, through the use of a numerical barrier method, these multiple constraints be removed. Therefore, if the ship path crosses an island domain, the time travelled objective is greatly penalized. This can be visually represented in figure (3).

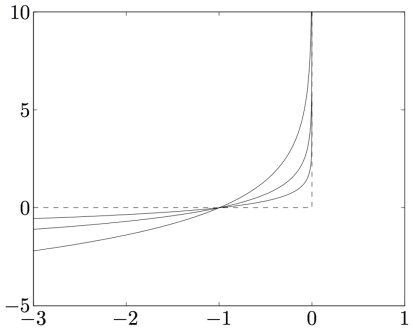


Figure 3: Barrier method refinement with increasing constant k

From equation (2.1) it can be seen that an increase in the r parameter steepens the penalization. Therefore it can be assumed that when $r = \infty$, the barrier approach is categorized nearly as a Heaviside step function with a value of infinity. Therefore, we can rewrite the previously determined barrier method expression more accurately as,

$$\tilde{f} = f + H(\mathbf{x}, g), \qquad H(\mathbf{x}, g) = \begin{cases} 0, & \text{for } \mathbf{x} \neq g,\ a < g < b \\ \infty, & \text{for } \mathbf{x} = g,\ a < g < b \end{cases} \tag{2.5}$$

Where, $H(n, m)$ is the Heaviside step function, **x** is the corresponding WayPoint design variable, and g is the relative the constraint within the domain. This barrier implementation allows for the model transformation of converting individual WayPoint constraints to a singular unconstrained optimization problem. This method allows the optimum to remain in the interior of the feasible domain. However, the consequence of this method is that the starting WayPoints need to be selected such that the location is within a feasible starting position. Implementing these obstructions will provide an additionally interesting analysis on the effects of the optimization routines and their corresponding responses.

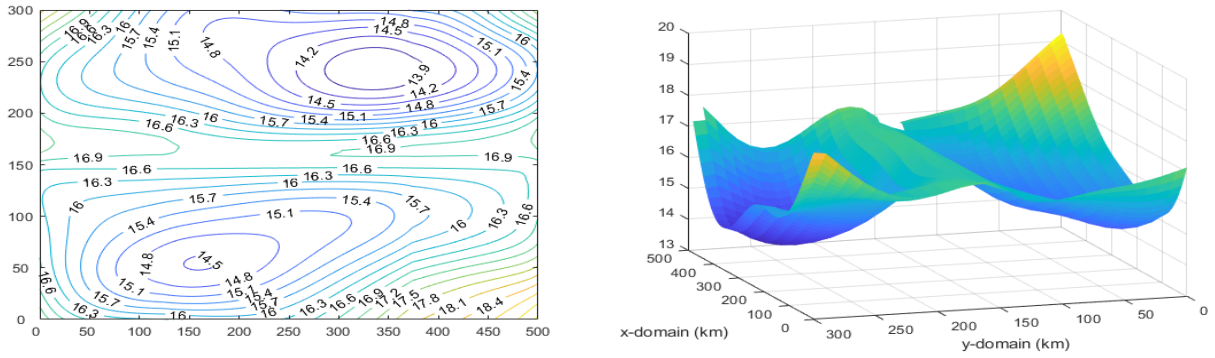
3 Initial Problem Investigation

In this section the initial problem is investigated. The problem is checked for boundedness, monotonicity, numerical noise and a sensitivity analysis is investigated. This problem analysis is done by means of a chosen WayPoint to be set on the domain. Using this, the WayPoint can be varied iteratively over all pixels making it possible to compute the objective function relatively easy. An example can be given by means of the scaling of the problem. The initial spatial domain involves a grid of 1500 pixels. With one WayPoint the objective surface can be plotted easily with a contour or surface plot. With two WayPoints this is already impossible. Next the problem scales roughly with 1500^n, with n WayPoints. So for three WayPoints more than a billion paths are possible. The Initial problem investigation is performed on a continuous domain with a linear interpolated path.

3.1 Boundedness

Boundedness indicates the domain of the objective problem. The function is constrained with respect to the problem boundary edge. The objective function was calculated by using one WayPoint and iterating over all the pixels. This will give the following plot.

From this plot it is easily identified that the function is bounded within the rectangular grid space of 500 km $\times$ 300 km. Note that clearly 2 possible optima lie within the given objective space.



(a) The contours indicate the travel time in hours

(b) 3D surface plot of the objective function

Figure 4: Objective function for one WayPoint. The function value of the objective function at a point corresponds with a WayPoint at that given point.

3.2 Convexity & monotinicity

In figure 4a the objective function with its two optima is plotted. From this we can estimate that the objective function is locally convex for values bounded by the contours $f_{obj} \leq 12.2$ a more clear indication would be if the first and second derivative of the objective function is plotted.

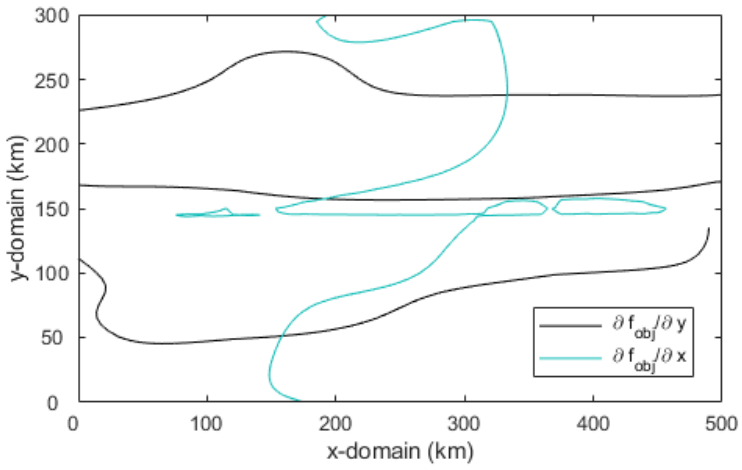
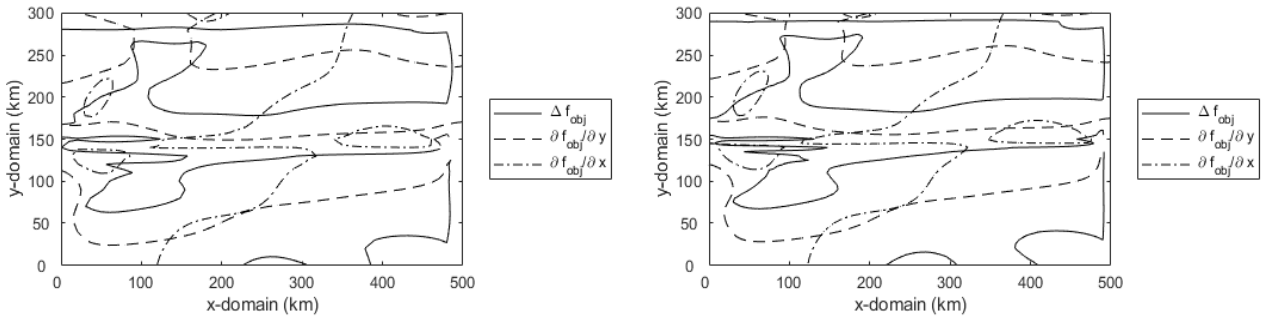


Figure 5: Iso lines where the derivative of the objective function with respect to the first and second derivative corresponds to 0

In figure 5, three extrema are identified where the isolines corresponding to $\partial f_{obj}/\partial x$ and $\partial f_{obj}/\partial y$ are equal to zero. Here, $|\nabla f_{obj}|$ is zero. Therefore an extreme must lie a these intersecting locations. From figure 4a we showed that two minima occur in this objective function, so the middle extrema must be a maximum. As such we can conclude that the objective function is convex in the two domains divided by the black isoline for $\partial f_{obj}/\partial y = 0$ in the middle.

in figure 6 the different gradients have been plotted for the objective function as well as the laplacian. The lines depicted are the isolines for when the gradient and the laplacian is equal to zero. When the laplacian is zero, the objective is either at an extreme or at a inflection point. When the derivative is zero the function is flat in the direction of the derivative. From this we can reason that the



(a) meshfactor = 1

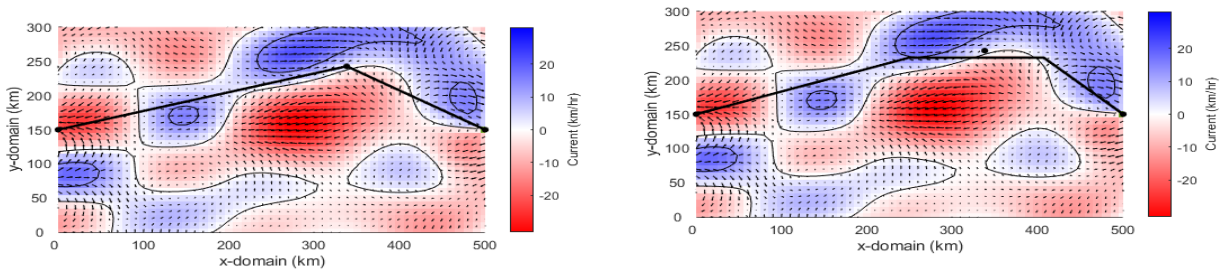
(b) meshfactor = 2

Figure 6: Iso lines for the laplacian and the different derivatives equal to 0

function is monotonic within the bounds of these isolines. The function is very non-monotonous in the full domain. Furthermore noisy behaviour can be seen around the point $[70, 110]$. This was justified when increasing the mesh size.

3.3 Numerical Noise

The numerical noise is determined to be dependent on two important factors. Firstly, there is the number of interpolation points to be used for the spline and secondly the gridsize can be varied as well. Using less grid and interpolation points equated to faster computational time at the cost of accuracy. Here, a trade-off has to be made. It is observed that with an increase in Pathpoints the solution converges to a solution. The minimum amount of pathpoints showed to be 3.

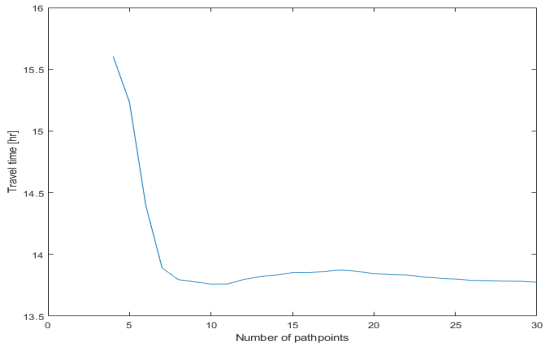


(a) Path corresponding to 3 pathpoints.

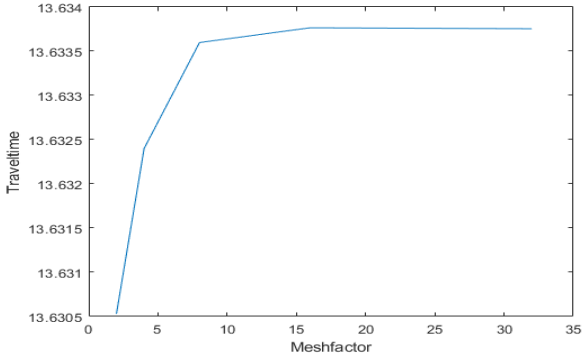
(b) Path corresponding to 4 WayPoints. Note that here the path does not go through the WayPoint

The variation in pathpoints and gridsize does not seem to show any noisy behaviour. It might seem that a large mesh factor is needed to ensure stable solutions for the optimum, but the difference between the high and low values is less than one percent. It was observed that for a really low mesh factor, no solution could be obtained. The lowest mesh factor possible was the base case of one. Lower than a factor of one would yield no results.

Overall, showing little dependency on the amount of gridcells and pathpoints it can be concluded that no extensive computational operations have to be made, i.e. the model is quick. By knowing that



(a) Noise dependence on Nr. of interpolation points. Note that the vertical does not start at 0



(b) Gridsize dependency on the travel time with a fixed WayPoint. Note the y-axis scaling doesn't start at zero

the derivatives do show dependency on the mesh size, the choice of first order optimization models is motivated.

3.4 Sensitivity Analysis

In this section a sensitivity analysis will be discussed. In order to find the sensitivity on the single objective with respect to a perturbation of a design variable, derivative information is required. As explained previously the optimization problem scales with $2n$, with n being the number of design variables. It could be argued that adjoint sensitivity analysis is a very good approach for this problem as this problem deals with one objective function and many design variables. The problem here however is that in the adjoint method a state is needed to compute the objective in order to solve the adjoint equation for. In other words we need a problem that can be formulated as

$$\langle Au, v\rangle = \langle u, A^{\dagger}v\rangle, \tag{3.1}$$

where A is the state matrix u would be the design variables and v the objective. On the right handside $A^{\dagger}$ is the adjoint state matrix. An important note is that deriving the adjoint equation for only possible for linear operations. In sensitivity or perturbation analysis this would be limited for a certain stepsize.

Because no information is present in the model on the state matrix, the adjoint formulation of the problem can not be derive. As such, we resort to a direct sensitivity approach by using finite differences. For each so-called WayPoint a perturbation is possible in both x and y direction. In this sensitivity analysis the norm of the gradient is studied as well as the effect of perturbing different WayPoints. Here the pitfall expresses itself in using the direct approach for sensitivity analysis. Consider the case where 5 WayPoints are used. Here the possible perturbations p of the objective function are

$$p = \sum_{i}^{5} \binom{5}{i} = 31. \tag{3.2}$$

Therefore, computing the (total) sensitivity scales badly with the number of WayPoints. In order to get a better understanding of the sensitivity, first the gradient of the objective is computed. The objective function is a function of the x and y coordinates of each way point. This can be written for a three WayPoint case as

$$f_{obj} = f(x_1, y_1, x_2, y_2, x_3, y_3) \tag{3.3}$$

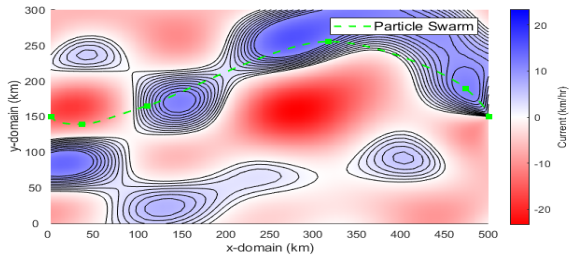
with the gradient

$$\nabla f_{obj} = \begin{bmatrix} \frac{\partial f}{\partial x_1} & \frac{\partial f}{\partial y_1} & \frac{\partial f}{\partial x_2} & \frac{\partial f}{\partial y_2} & \frac{\partial f}{\partial x_3} & \frac{\partial f}{\partial y_3} \end{bmatrix}^T \tag{3.4}$$

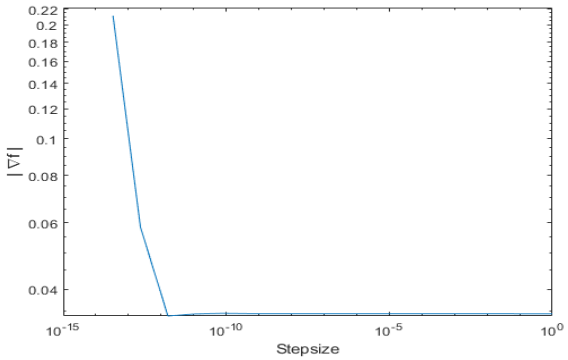
The gradient is computed in a discrete way by using a central finite difference method

$$\frac{\partial f}{\partial x_i} \approx \frac{f(x_i + h) - f(x_i - h)}{2h} \tag{3.5}$$

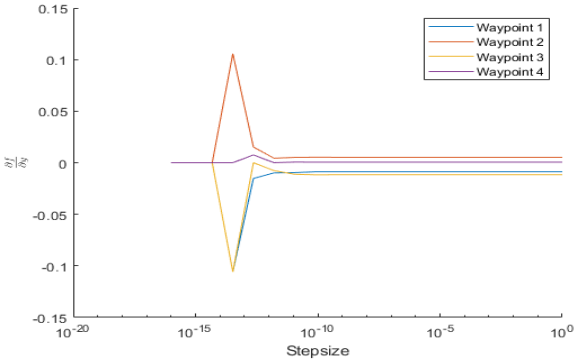
This is done for a range of stepsizes as can be seen in figure **??**. The overall objective function is not very sensitive with respect to the used stepsize. For a small stepsize it can be seen that the condition error starts to dominate the gradient it was observed. In decreasing the meshsize, the gradient was observed to increase proportionally, but the same trend was kept. For a large mesh size the derivative was no longer observed to give local information anymore, hence the sensitivity is plotted up to a mesh size of 1.



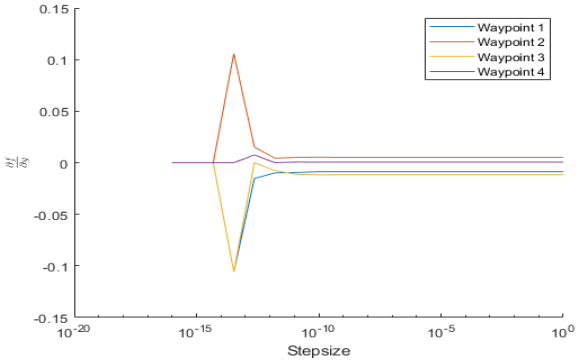
(a) Path on which sensitivity is studied



(b) The norm of the gradient with respect to the stepsize



(c) $\partial f / \partial x$ with respect to the stepsize



(d) $\partial f / \partial y$ with respect to the stepsize

4 Initial Optimization on Simplified Problem

4.1 Motivation of optimization approach/choices

From the initial problem investigation as seen in section (3), it can be clearly determined that the subsequent objective function is non-convex in a non-convex feasible domain. As such the optimality criteria for determining a global minimum to the path minimization problem is not possible. As such, the motivation of the initial investigation will be to work with a simple two-variable optimization problem (i.e. single WayPoint). This simplicity will allow for a clear representation of the objective function and expected global optimal results. Therefore, an approach of using different optimization algorithms will be implemented to determine which optimizer works best in the given non-convex situation.

4.2 Optimization algorithm description and parameter selection

Three different optimization algorithms will be investigated in the following analysis. These algorithms were each chosen to evaluate the effectiveness of a black-box simulation solution analysis in search of a global optimum when the function details are not well known in advance.

The first approach was a multi-start local optimizer using the build-in MATLAB function 'fminunc'. This build-in function contains a 1^{st} order Quasi-Newton BFGS algorithm, which are designed for a unconstrained gradient based-problem. These algorithms are efficient in finding an optimum; however, the found optimum depends on the initial conditions of the optimization problem. With the same initial conditions, the optimizer always finds the same optimum. Unless the objective function is convex, this optimum does not have to be the ability to determine global optimum. To have a higher chance at finding the global optimum, the optimization is completed a multiple times with different initial conditions. The second method is through the incorporation of direct search 0-order stochastic methods. These are generally very robust and can also work for discontinuous or non-differentiable functions. The greatest strength of these methods are that they have the potential to find the global optimum amongst a objective with many local minima. Unfortunately, these methods become less attractive for high variable counts (<10). Two of these methods will be investigated; a biologically inspired method known as Particle Swarm Optimization, and a nature inspired self-written Simulated Annealing Algorithm. Each method incorporates a degree of randomness in their approach to allow for a chance of finding a local minimum. All three of these algorithms will be further explained in the coming section analysis.

4.2.1 Quasi-Newton BFGS Optimization

Here a Quasi-Newton method is discussed in finding the optimum path. As was shown in section 3.2 where convexity and monotonicity was analysed, second order gradient information did not prove to be all too reliable. A standard Newton method requires a Hessian matrix to find the root of the objective gradient. Furthermore, to find an optimum, the domain has to be convex. As such, using this method is only useful when no barriers are present and some form of randomization is used. A good choice would be to find a global optimum crudely with a 0^{th} order method after which a refinement can be made by using a newton method.

A solution which is often not able to directly calculate the Hessian matrix is to use a Quasi-Newton approach. A Quasi-Newton method approximates the Hessian by using only first order derivative information as it uses only the first order approximation of the gradient.

$$\nabla f_{k+1} - \nabla f_k = \tilde{\mathbf{H}}(\mathbf{x}_{k+1} - \mathbf{x}_k) \tag{4.1}$$

A substitution,

$$\tilde{\mathbf{B}}_{k+1} = \tilde{\mathbf{B}}_k + \Delta\tilde{\mathbf{B}}_k \tag{4.2}$$

$$\tag{4.3}$$

will be the approximation of the inverse of the Hessian. Here, different update algorithms can be used. In the optimization routine the choice is made for the conjugate gradient BFGS update schemes. This scheme is of rank 2, which are regarded as the best general-purpose solution for unconstrained (hill-climbing) optimization.

4.2.2 Particle Swarm Optimization

Partical swarm is a 0^{th} 'brute' force order method where a large number of random initial conditions are sampled over the domain.

Particle swarm optimization works by simulating herd or swarm behaviour. Here the trend is that each 'individual' has a individual behaviour and a swarm behaviour. The swarm behaviour makes sure that the particle moves in the optimal direction by comparing its position with the particle that has the minimal position. It then sets an update 'speed' in the direction of the global optimum. The individual behaviour is simulated based on the particle's historic minimum. This individual speed is in the direction of the historic minimum.

$$\mathbf{v}_{i+1} = \mathbf{v}_i + c_1\mathbf{r}_1 \otimes (\mathbf{y}_i - x_i) + c_2\mathbf{r}_2 \otimes (\mathbf{Y}_i - x_i) \tag{4.4}$$

$$\mathbf{x}_{i+1} = \mathbf{x}_i + \mathbf{v}_{i+1} \tag{4.5}$$

The c scalars are control parameters for the optimization algorithm. The **r** vectors are random number vectors between 0 and 1 to introduce some randomness direction to the problem. Increasing the c_2 parameter with respect to the c_1 tends to increase the speed in the direction of the swarms optimum. This might seem favorable as this leads to fast convergence, but this also increases the possibility in overshooting a optimum. This can occur especially in problems involving multi-modality with large gradients. A too large c_1 parameter makes sure that there is essentially no swarm behaviour. This makes sure that a balanced choice between the two variables must be taken.

Another aspect in swarm optimization is that sufficient samples must be taken such that all the possible paths are initially taken. It can be argued that as the problem size increases, more samples have to be taken which would influence negatively on computation time. An advantageous property of the different samples is that each velocity computation can be calculated in parallel before the global swarm minimum has to be passed to each particle. The main advantage in the particle swarm approach is that a global optimum can be found in a discontinuous domain as the initial swarm will be spread out over the domain if the size of the swarm is large enough. The swarm size is chosen to be 100.

4.2.3 Simulated Annealing Optimization

The simulated annealing process is a random-physical process inspired optimization routine. Annealing refers to the thermodynamic process of heating a solid and then cooling it slowly. Atoms then assume a nearly globally minimum energy state. The algorithm simulates a small random displacement of an atom at a temperature that results in a change in energy. If the change in energy is negative, the energy state of the new configuration is lower, and the new configuration is accepted. If the change in energy is positive, the new configuration has a higher energy state; however, it may still be accepted according to the probability factor:

$$P_{accept, f(y)>f(x)} = e^{(\frac{-\Delta E}{T})} \tag{4.6}$$

Where T is the current temperature, and ΔE can be described as the change in the objective functions per new random perturbation step with respect to the previously determined average function output [2]. This allows for a relative change in the objective function and can be further described as,

$$\Delta E = \frac{f(y) - f(x)}{f_{avg}} \tag{4.7}$$

In this case it can be seen directly that the probability is proportional to temperature; as the solid cools, the probability gets smaller. However, it is also inversely proportional to change in energy; as

the change in energy is larger the probability of accepting the change gets smaller [1]. To ensure an equal reduction of temperature per iteration is obtained, the following fraction can be implemented,

$$T_{Frac} = \left(\frac{T_1}{T_n}\right)^{(\frac{1}{n-1})} \tag{4.8}$$

Where, T_n is the final iteration temperature. If through the iteration procedure, the objective is lower, the new design is made the current design; if it is higher, it may still be accepted according the probability given by the probability factor. At this point, the probability is compared to a random number drawn from a uniform distribution between 0 and 1; if the random number is smaller than the probability, the configuration is accepted. This allows the algorithm to escape local minima. Although the algorithm is not guaranteed to find the best optimum, it will often find near optimum designs with many fewer design evaluations than other algorithms. While the method is relatively simple to implement, it can still be computationally expensive [1].

Test Function Analysis To evaluate whether the self-scripted simulated annealing optimization algorithm performs as expected a test function analysis was completed. This study applies the optimization algorithm to a predetermined artificial landscape to determine optimization characteristics such as, convergence rate, precision, robustness, and/or general performance. Since the overall objective function is non convex with multiple minima, the function will also include multiple optimal points in the domain. The simulated annealing, if scripted appropriately, should be able to overcome these less optimum points in search of the global minima. This test function can be described using the following expression,

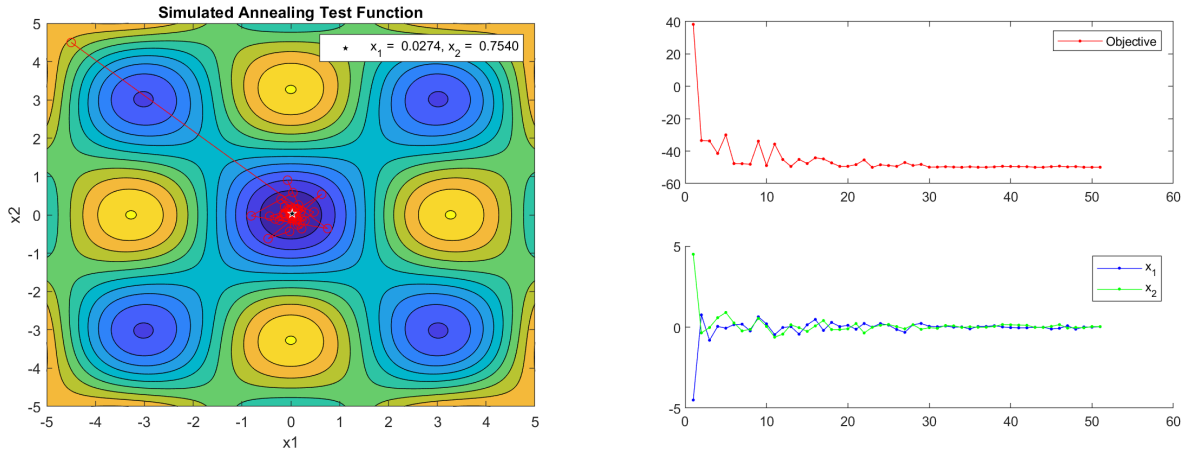
$$f_{test} = -50 \cdot cos(x_1) \cdot cos(x_2) + x_1^2 + x_2^2 \tag{4.9}$$

Where, x_1 and x_2 are arbitrary design variables. It should be noted that the test function's global optimum point is located at position (0,0). The implementation of the Simulated Annealing algorithm as well as the test function landscape can be seen in figure (10a). The program has the initial settings for the number of iterations per step and the number of generated sample points per step set equally. The probabilities are also established with a high likelihood of accepting a worse solution early in the procedure. The full properties can be seen highlighted in table (1).

Table 1: Initial simulated annealing parameters

number of iterations	n	50
number of trials per iteration	m	50
P_1	-	0.70
P_{50}	-	0.001

Based on the results observed, the self script SA program works quite nicely. The start position was changed to consider multiple initial conditions. However, the current settings resulted in a very consistent solution where the final result was found nearly every time. Figure (10b), highlights the objective solution, and design variable selection process. It can be observed that near the end of the run, the probability became low enough that worse solutions were no longer considered as opposed to the beginning. In this region, the algorithm has a high degree of freedom to randomly explore the design space and gradually locate the global optimal location. This is highlight in the lower section of the figure, where the design points can gradually be seen to converge upon the final optimal solution at (0,0).



(a) Test function and results

(b) Objective and variable changes per iteration

Figure 10: Self-written simulated annealing test function analysis

From this test function analysis, it can be confirmed that the simulated annealing self-script works relatively well when confronted with a non convex domain. The overall convergence rate was relatively quick, at a time of 0.0159s. The final precision is also quite impressive through the determination of the final points $x_1 = 0.027$ and $x_2 = 0.0318$. While this is not exact it is still a very close approximation. With the success of the test function analysis, the self-scripted SA will be further compared with MATLAB built-in simulated annealing, along with the other algorithms on the actual problem domain.

4.3 Comparison of Results and Interpretations

The results of the simplified two-variable problem (SP) for each optimization algorithm are first computed for an linear route. A table with a summary of the optimized travelling times for each algorithm can be seen summarized in table (2). A corresponding visualization of the optimized path can be seen in figure (11) as well. This figure highlights each optimization route, and associated WayPoint placement. It should be noted that there are two simulated annealing algorithms investigated. The first, is the self-written script previously discussed. Whereas, the second is the built-in MATLAB function, 'simulannealbnd' with its default settings applied. This will help provide a baseline comparison between the two.

Table 2: Simplified Problem w/ Linear Path Optimization Summary

Optimization Algorithm	Travel Time	Time Reduction	Rank
Linear (x = 250km & y = 150km)	16 hours, 39.5 minutes	n.a.	5
Quasi-Newton BFGS	14 hours, 28.3 minutes	2 hours, 11.2 minutes	4
Particle Swarm	13 hours, 37.8 minutes	3 hours, 1.7 minutes	**1**
Simulated Annealing (MATLAB)	13 hours, 38.0 minutes	3 hours, 1.5 minutes	2
Simulated Annealing (Self Script)	13 hours, 39.0 minutes	3 hours, 0.5 minutes	3

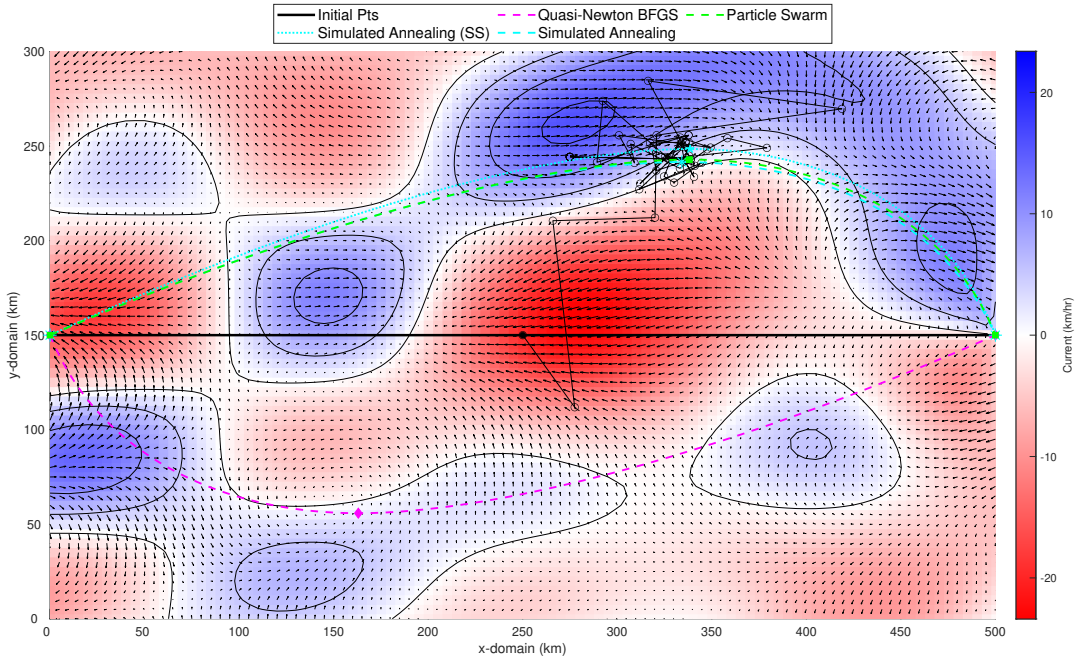


Figure 11: Simplified optimization algorithm comparison with point at x=250km & y=150km (n=2)

4.3.1 Quasi-Newton BFGS

Based on the results the modified 1st order Quasi-Newton algorithm provides the smallest reduction in time optimization. Based on figure (4a), it can be directly seen that the optimization algorithm unfortunately falls within a local minima region. The BFGS algorithm is a hill climbing algorithm and therefore struggles with the multi-modality of the problem. The algorithm searches the direction of the steepest descent with a high chance of convergence to a local optimum. This determination of an optimum can be checked through an optimaltiy evaluation. The conditions for local minimum of unconstrained problems are,

$\nabla f = 0$	First Order Necessity Condition
H positive definite	Second Order Sufficiency Condition

The corresponding MATLAB first order optimality is firstorderopt: $1.2912e{-}05$. Since this numerical value is very near 0, the first condition can be considered met. Next the hessian is analyzed using Sylvester's Rule to determine positive definiteness,

$$\mathbf{H}_{QN} = \begin{bmatrix} 0.0064 & -0.0024 \\ -0.0024 & 0.0064 \end{bmatrix} \longrightarrow \begin{cases} 0.0064 > 0 \\ (0.0064 \cdot 0.0064) - (-0.0024 \cdot -0.0024) > 0 \end{cases} \tag{4.10}$$

Since, all determinants of **H** and its principle sub matrices are positive the second optimaity condition is also met. Therefore, it can be determined that an local optimum is determined.

4.3.2 Particle Swarm

The particle swarm optimization shows the best reduction in time for the simplified problem. With an initial swarm sie of 100 the optimum is obtained. The options for the particle swarm in MATLAB are quite extensive. These are kept mostly default. When the optimization is run with islands and more WayPoints, the choice can be made to run the optimization in parallel. It was observed that for a one WayPoint problem no significant speed gain was obtained

```
Default properties:
                CreationFcn: @pswcreationuniform
          FunctionTolerance: 1.0000e-06
               InertiaRange: [0.1000  1.1000]
            InitialSwarmSpan: 2000
               MaxIterations: '200*numberofvariables'
          MaxStallIterations: 20
                MaxStallTime: Inf
                     MaxTime: Inf
       MinNeighborsFraction: 0.2500
             ObjectiveLimit: -Inf
       SelfAdjustmentWeight: 1.4900
     SocialAdjustmentWeight: 1.4900
                  SwarmSize: 'min(100,10*numberofvariables)'
```

When the optimum is satisfied MATLAB gives the output

```
    Optimization ended: relative change in the objective value
    over the last OPTIONS.MaxStallIterations iterations is less than ...
        OPTIONS.FunctionTolerance.
```

The default MaxStallIterations is 20. This indicates the number of iterations since the last change in the best solution indicated by $\mathbf{Y}_i$ in section 4.2.2. From the output it can be concluded the program will be terminated if

$$\left| \frac{\mathbf{Y}_{i,old} - \mathbf{Y}_{i,new}}{\mathbf{Y}_{i,old}} \right| < tol \tag{4.11}$$

or if the failsafe of $200 \times$ numberofvariables is reached.

4.3.3 Simulated Annealing

The simulated annealing self-scripted algorithm successfully, overcomes the local optima region, and is able to find the global minimum region. Unfortunately, the exact minimum is not determined, however the location of the optimized WayPoint provides a very good first estimate. When compared with the built-in MATLAB simulated annealing algorithm 'simannealbnd', the self-script determines a solution which is less slightly less accurate.

Parameter Comparison The amount of iterations set for the written optimization algorithm in this case is 100 iterations. Within each iteration a total of 100 trial evaluations were completed. Therefore, the total function evaluations which occur in the algorithm is 10,000. Furthermore the start and end probability are 0.7 and 0.001 respectively. This entire process required an evaluation time of 10.3373s to complete. whereas, for the built-in function the output displays;

```
outputSA = iterations: 197,
           funccount: 1999
           message: 'Optimization terminated: change in best function value less ...
               than options.FunctionTolerance.'
           totaltime: 2.6334
```

It can be seen that the amount of iterations completed were roughly equivalent, however the total function count is much lower. This would roughly be equivalent to 10 trial evaluations per iteration. Even with such a dramatic reduction in function evaluations, the built-in function performs better than the self-script. It should be noted that default settings were used in the MATLAB function which may have a dramatic impact on the final solution. The main adjustable properties can be seen listed below,

```
    Default properties:
          FunctionTolerance: 1.0000e-06
       InitialTemperature: 100
     MaxFunctionEvaluations: '3000*numberOfVariables'
              MaxIterations: Inf
       MaxStallIterations: '500*numberOfVariables'
```

The stopping criteria in this evaluation indicates that the function fell below the set tolerance of FunctionTolerance: $1.0000e-06$. Therefore, it can be concluded that while the self-script, does as intended, the MATLAB function is more robust, faster, and generally more accurate as a higher degree of control and evaluation is possible.

4.3.4 Effects of the Initial Starting Condition

While, a thorough analysis is done for a linear path, the influence on the initial WayPoint starting position must be further explored. As such, various starting positions were investigated along with resulting optimized travel times. The effects were quite clear, as such the results of a single case where the global minima region is determined for all algorithms can be seen summarized and table (3) and visualized in figure (12).

Table 3: Simplified Problem with x = 50 & y = 250 Path Optimization Summary

Optimization Algorithm	Travel Time	Time Reduction	Rank
x = 50km & y = 250km	15 hours, 59.4 minutes	n.a.	5
Quasi-Newton BFGS	13 hours, 37.8 minutes	2 hours, 21.6 minutes	**1**
Particle Swarm	13 hours, 37.8 minutes	2 hours, 21.6 minutes	**1**
Simulated Annealing (MATLAB)	13 hours, 37.9 minutes	2 hours, 21.5 minutes	3
Simulated Annealing (Self Script)	13 hours, 40.0 minutes	2 hours, 19.4 minutes	4

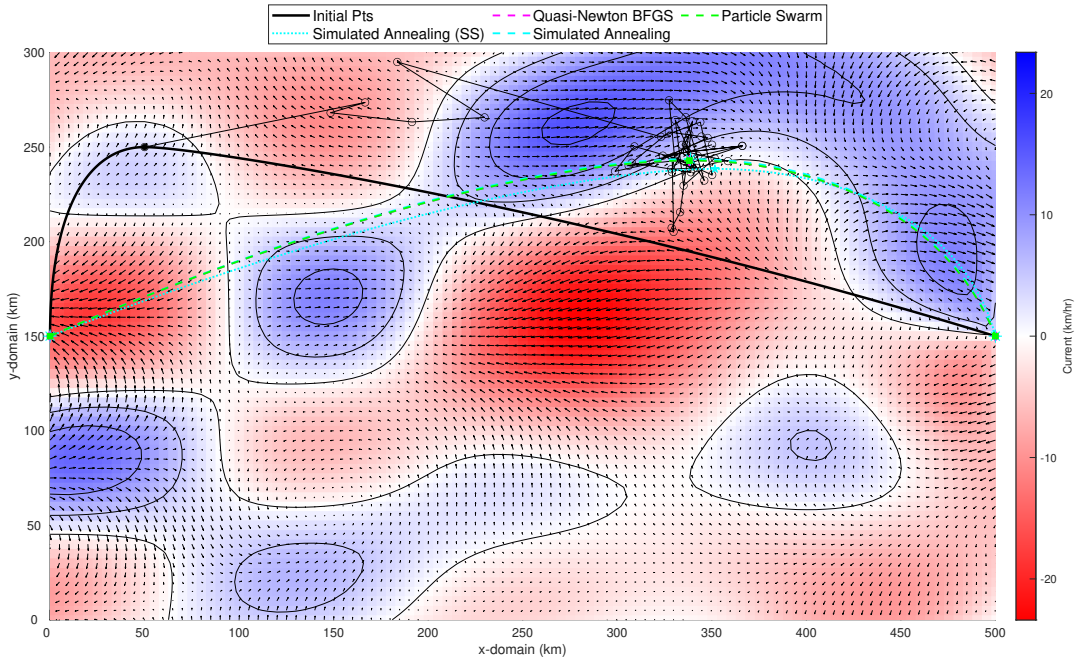


Figure 12: Simplified optimization algorithm comparison with point at x = 50 km & y = 250 km (n=2)

From the analysis, and the results it can be determined that the algorithms which have randomness properties, are not easily effected by the initial starting conditions. These include both simulated annealing, and particle swarm results. However, the Quasi-Newton BFGS algorithm is greatly influenced. From the results it can be seen that not only does the algorithm find the correct region, it also determines the greatest reduction in travel time. These effects were observed, in almost all cases when the initial condition fell within the monotonically decreasing regions. Since this algorithm is an approximate first order method, the use of gradients allow for a very good approximation when conditions fall in a region of convexity for both the objective and feasible domain. This allows for a clear determination of a local minimum, as the optimally conditions can be clearly evaluated. However, since we know the domain lies in the global minimum region, the only optimality condition which needs be evaluated is,

$$\nabla f = 0 \tag{4.12}$$

In this case the output for the first order optimality condition can be determined as, firstorderopt: $2.94e{-}7$. While not entirely zero, the stopping condition can be considered met as set optimality tolerance of 1.0e-6, is exceeded.

5 Optimization of Actual Problem

5.1 Motivation of optimization approach/choices

Having established a detailed analysis of a simplified, two variable optimization problem, a more advanced situation will be analyzed. This section includes the implementation of three WayPoints and five barrier obstructions.The additional way points will introduce a total of 6 continuous variables to the

problem. A stated previously, the problem can become transformed from a constrained optimization problem to an unconstrained problem by implementing the barrier methodology. Furthermore it was shown that the problem for only one way point was highly multi-modal and second order derivatives proved hard to obtain. This motivated the use of primarily 0^{th} order methods.However, it should be noted that the number of points were chosen such that the variables did not exceed a total of 10, as this can cause greatly reduced accuracy and efficiency within these method types.

Additionally, a hill climbing method makes use of C^1 or C^2 continuity. As this is clearly not the case in this barrier problem, an approximation has to be made in approximating the Jacobian and Hessian matrices for finding optimums. In choosing a 0^{th} order method it is more easy to 'scan' the domain for different convex regions. Next to that, less function evaluations are needed. In a 1D problem, evaluating a gradient requires one extra function evaluation or more. Therefore, the inclusion of additional points and a semi-discontinuous domain will prove to be a interesting investigation of which optimization algorithms fair the best. It should be noted, the due to the increased complexity and outcome possibilities, the analysis will use a more qualitative comparison approach between the selected optimization algorithms.

5.2 Investigation of obtained optimum

The optimization for the multi-WayPoint problem with islands is discussed in this section. The initial conditions vary somewhat from the problem without islands. due to the barrier method, no solution can be obtained if the path goes through the islands. Therefore, an initial path must be selected such that it does not travel through the islands as depicted in figure 13. Depending on the number of islands the possible paths can vary quite extensively. As such, results are observed for two different random paths directions through a total of 5 implemented islands. These results can be seen summarized in both tables (4), and (5) as well as figures (13) and (14) respectively.

Table 4: Advanced Problem with Upper Random Path Optimization Summary

Optimization Algorithm	**Travel Time**	**Time Reduction**	**Rank**
Upper Random Selected Path	18 hours, 46.7 minutes	n.a.	5
Quasi-Newton BFGS	14 hours, 33.9 minutes	4 hours, 12.8 minutes	4
Particle Swarm	13 hours, 39.9 minutes	5 hours, 6.8 minutes	**1**
Simulated Annealing (MATLAB)	13 hours, 43.4 minutes	5 hours, 3.3 minutes	2
Simulated Annealing (Self Script)	13 hours, 51.9 minutes	4 hours, 54.8 minutes	3

Table 5: Advanced Problem with Lower Random Path Optimization Summary

Optimization Algorithm	**Travel Time**	**Time Reduction**	**Rank**
Lower Random Selected Path	17 hours, 3.0 minutes	n.a.	5
Quasi-Newton BFGS	15 hours, 5.4 minutes	1 hours, 57.6 minutes	4
Particle Swarm	13 hours, 43.5 minutes	3 hours, 19.5 minutes	**1**
Simulated Annealing (MATLAB)	13 hours, 56.5 minutes	3 hours, 6.5 minutes	3
Simulated Annealing (Self Script)	13 hours, 52.9 minutes	3 hours, 10.1 minutes	2

Results show that the 0^{th} order particle swarm algorithm proves to be the most suitable to handle such a situation. Next is the MATLAB based simulated annealing, then the self scripted simulated annealing and lastly the BFGS. Ultimately, its appears as though the random 0^{th} order methods still handle the more advanced case relatively well, as it consistently seems to find a minima region. Between the two simulated annealing algorithms, not only is there a difference in performance but

also in an optimum when scaling to more variables. This performance difference is not unambiguous. From two runs it is hard to distinguish a bias towards one method over the other. Due to restrictions in time it was no longer possible to study this much further. However, it should be noted that while the self-script performs admirably, heavy tuning of the control parameters is required to have consistent results as opposed to the MATLAB algorithm Thus, the MATLAB simulated annealing was found to be both more robust and faster in general when solving the advanced optimization problem.

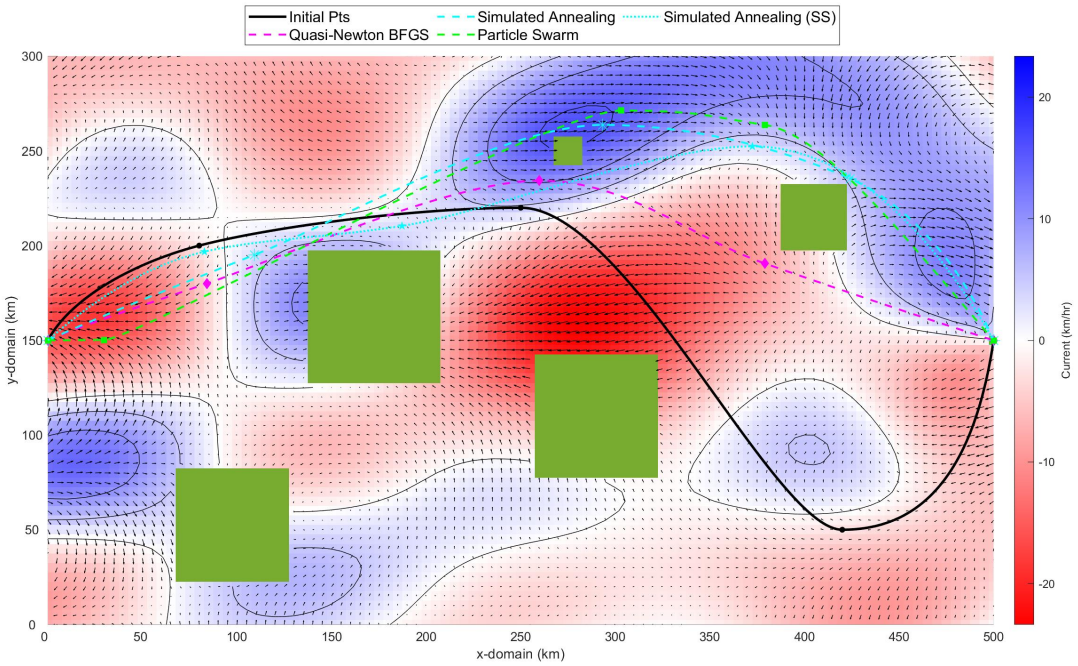


Figure 13: Advanced optimization algorithm comparison with island obstructions and upper path (n=2)

The BFGS shows problems in finding the optimum. It can be observed that this algorithm does not have the capabilities to overcome the barriers and instead finds its optimal route restricted with the obstructions. Thus, these 1st-order descent algorithms are a cause for concern when used in such cases. However, if we assume the Particle Swarm did find the global optimum region, and the initial path is chosen such that the problem does not suffer from any barrier (i.e. optimum is bounded from the same limit direction), then the BFGS generally gives an equally or more accurate optimum path. Therefore, when no islands are used, agreement is observed between PS an BFGS which are both the fastest paths in this case. When islands are introduced, performance of the BFGS is diminished and greatly relies on the initial starting WayPoint positions. Unfortunately, these properties are unfit for most 1^{st} and 2^{nd} order methods.

6 Overall Conclusions and Recommendations

An spatial 2D path optimization problem has been formulated by means of a fitted polynomial through n WayPoints, which results in $2n$ degrees of freedom. The objective was to minimize the travel time from point A to point B. In section 3 the problem's boundedness, continuous properties, and noise behaviour have been analyzed. Furthermore a sensitivity analysis has been done on evaluating the

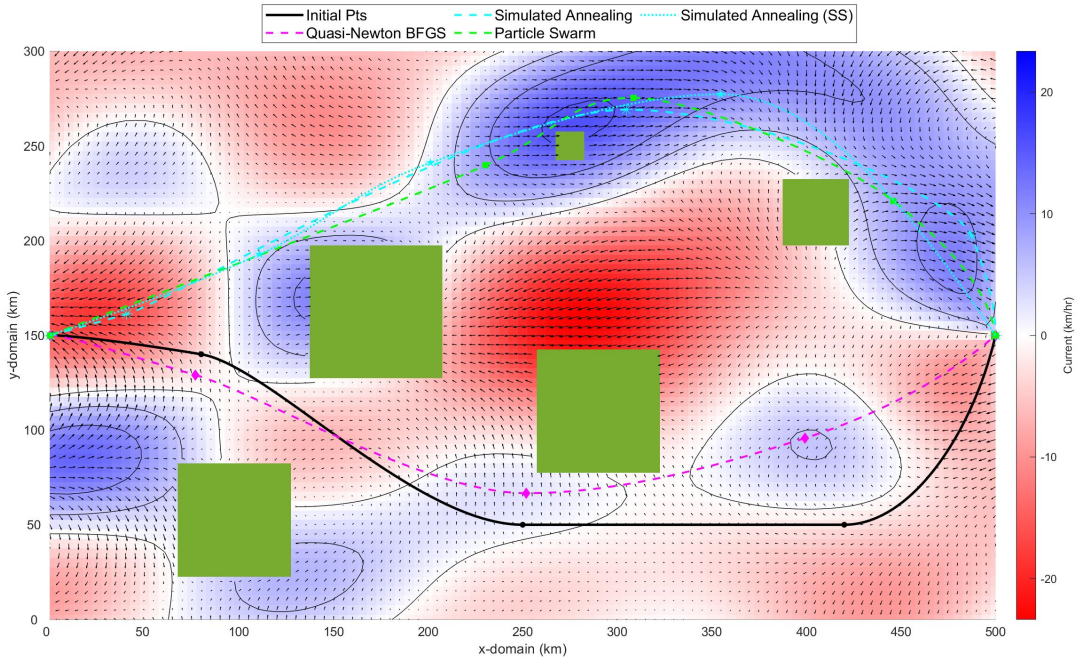


Figure 14: Advanced optimization algorithm comparison with island obstructions and lower path (n=2)

gradient of the objective function. This section motivated the use of primarily 0^{th} order methods. To further disclose the use of 0^{th} over 1^{st} methods various 0^{th} order methods have been tested on a simplified problem in section 4. The robust 1^{st} order BFGS optimization method performed very well in this limited approach, as optimality conditions are clearly met each run. However, when additional WayPoints and obstructions (i.e. barriers) are introduced to replicate a more advanced and realistic problem, the deficiencies of the 1^{st} order methods became transparent. Here the BFGS performed worse than the 0^{th} order random methods, where the main cause was determined to be the inability to overcome any barrier. Based on the final results, it can be concluded that the Particle Swarm optimization algorithm was the consistently most accurate procedure throughout the analysis. As such the authors therefore advise to search for the global optimum by using Particle Swarm optimization, after which refinement of the optimum can be done using Quasi-Newton with a BFGS Hessian update scheme.

While the overarching goal was only to observe the effects and become familiarised with optimization and the various methodologies associated, a few recommendations to improve the study cam be suggested. The first is improvement of the implemented barrier method. The approach was done numerically to penalize the objective function in the form of a step function approximation. However, This effect does not allow success for gradient methods as no obstruction gradients are present in the feasible domain . Therefore further improvement to allow for a smoothing effect on the barrier would be a welcome and interesting addition to see the effects on the 1^{st} order optimization algorithms. Secondly, while a qualitative approach was considered for the advanced problem, a deeper look into output results such as optimality criterion, parameter setting effects, and model point/mesh refinement will further expand insight into the acquired results and associated changes. Furthermore, model improvements such as implementation of real-world weather data, current data and topographical maps can greatly increase the validity of real world application.

Bibliography

[1] P. J. D. Hedengren. Simulated annealing tutorial design optimization, 1999. URL http://apmonitor.com/me575/index.php/Main/SimulatedAnnealing.

[2] N. Leite, F. Melício, and A. C. Rosa. A fast simulated annealing algorithm for the examination timetabling problem. *Expert Systems with Applications*, 122:137 – 151, 2019. ISSN 0957-4174. doi: https://doi.org/10.1016/j.eswa.2018.12.048. URL http://www.sciencedirect.com/science/article/pii/S0957417418308169.

A MATLAB Script

Due to the size and length of the MATLAB model script, it will only be included with the final submission of the completed report. The MATLAB package will include, the full model script (includes the simulated annealing self-script), as well as all associated functions to determine optimal travel path.