Delft University of Technology
Faculty Electrical Engineering, Mathematics & Computer Science
Bachelor of Science in Applied Mathematics

An airborne wind farm

Optimising the design of an offshore wind farm using airborne wind energy systems

Tutor: Marieke Kootte
Author: Tim Bosman

Delft, The Netherlands
April 2025

TU Delft

Nomenclature

α	Entrainment constant Park model
ρ	Density
A	Rotor area
A_{wing}	Wing area
C_D	Drag coefficient
C_L	Lift coefficient
C_p	Power coefficient
D	Outer diameter of flightpath
d	Inner diameter of flightpath
D_w	Outer wake diameter
d_w	Inner wake diameter
E	Entrainment constant Kaufman-Martin model
P	Wind power
R	Rotor radius
u	Wind velocity
u_∞	Initial wind velocity
u_r	Wind velocity directly behind rotor
u_w	Wind velocity in wake
x	Distance in downstream direction
z	Hub height
z_0	Surface roughness

Abstract

In this research, the feasibility of airborne wind energy systems (AWES) in wind farms was investigated. Thereafter an optimal layout for a wind farm using AWES was constructed. The EU seeks to have carbon-neutral power generation by 2050, for which wind energy could be a viable solution. Conventional wind turbines, however, use up a lot of resources. An alternative in AWES is considered because of their larger power-to-mass ratio.

First two models are introduced. A modified version of the Park model by Jensen (1983) and the Kaufman-Martin model by Kaufman-Martin et al. (2022). These were then used to predict the wake effects of AWES. To compare them to reality, the Offshore Windpark Egmond aan Zee (OWEZ) was replicated. Here, instead of the conventional wind turbines of model Vestas 90-3MW, AWES of model Makani M600 were placed. It was found that the wake effects generated by the Makani M600 were significant and not negligible, as previously thought. The wind was fixed to come from one direction where the systems were completely submerged in the wake of its downstream component. Here the power-to-mass ratio of the replicated OWEZ was 3.9 times higher than that of the OWEZ.

Using the same dimension of the OWEZ, a genetic algorithm was used to find an optimal layout for an AWES wind farm. The Park model was used for this algorithm. In the first simulation the algorithm was allowed to place the Makani M600 systems freely across the area. For the second simulation the placement was grid-constrained, to correspond to real-life situations more. It was found that the power-to-mass ratio was 5.7 and 5 times higher than that of the OWEZ respectively. Therefore, in conclusion, the wake effects of AWES are considerable and because of its higher power-to-mass ratio it is preferable to use AWES instead of conventional wind turbines for future development.

Contents

1 Introduction

The future of energy generation lies in renewable energy, with wind power being its most efficient energy source[1]. Offshore wind energy in particular is a popular energy generation method as it uses otherwise unused space. More importantly, the wind flow is more homogeneous offshore, due to the absence of tall buildings, hills or trees. These wind turbines are then clustered together in wind farms for cost efficiency.

Since wind turbines extract energy from the wind, the wind behind the rotor has a lower energy content. This results in a turbulent wind with a lower velocity[2]. This downstream wind is called the wake of a turbine[2]. The wind turbine power equation is dependent on the wind velocity. Hence, a lower wind velocity results in a lower energy extraction. The wake of upstream wind turbines therefore, has a negative influence on the energy extraction of downstream wind turbines. These negative effects are called wake effects. Barthelmie et al. (2010) state that in conventional wind farms, downstream rows of wind turbines generate up to 40% less energy than the first row of wind turbines [4]. Therefore, it is important to place the wind turbines strategically to minimise the wake effects in the wind farm and optimise energy production. Conventional wind farms make use of horizontal-axis wind turbines (HAWTs).

In 2023 all of the offshore wind farms had a capacity of 67GW[5], which, if used to its full potential, made up only 2% of the global energy demand[6]. However, the EU seeks to be carbon neutral by 2050. With the amount of material it uses, scalability towards net-zero carbon emissions becomes a problem when only using the current technology. To solve this, new technologies with higher power-to-mass ratios such as airborne wind energy (AWE) emerge[36]. AWE is the overarching name of concepts that convert wind energy into electricity, most commonly using autonomous kites or aircraft attached to the ground by a tether[7]. These systems used to convert wind energy to electricity are called airborne wind energy systems (AWES).

Wake effects weren't considered significant for AWES. Due to the large swept area of the airborne wind energy systems and the small size of the airborne devices, the wake effects were often ignored[8][9]. Therefore recent studies, such as Malz et al. (2018) and Roque et al. (2020), studying airborne wind energy (AWE) farms did not consider wakes[10][11]. Recent studies such as [23][12][14][13][49][15] have shown that the wake effects of AWES are considerable and therefore are not to be neglected.

This research focuses on the research question: *What is the feasibility of airborne wind energy systems in a wind farm and what would be an optimal layout for such a wind farm?* To answer this question, first models used to describe the effects of wakes are discussed in section 2. Thereafter, the concept of AWES itself is discussed in 3. In section 4, two models are derived to predict the wake effects of AWES, which are later implemented in section 5. In 5.1, the Offshore Windpark Egmond aan Zee (OWEZ), which uses conventional wind turbines, is replicated using AWES. Thereafter, in 5.2, an optimal layout for the AWES wind farm is found. Lastly the results of section 5 are discussed in section 6.

2 Wake models

Wakes can be divided into two regions: near wakes and far wakes. Near wakes are the area approximately up to two rotor diameters (2D) behind the rotor [16]. Here the properties of the rotor are distinctly recognisable and of influence in the flow field: the shape of the blade directly affects the wake. About two to five rotor diameters (2D-5D) downstream is called the transition region. The region beyond this is called the far wake. In the far wake the focus lies on the influence of wind turbines in wind farms, rather than individual properties of the actual rotor. The properties of the rotor here are only of influence in two ways: a wind velocity deficit and an increase in turbulence intensity[17]. During this research, the effects of the near-wake are not considered as they are not of interest to the modelling of wind farms.

Using the Navier-Stokes[1] (NS) equations, it is possible to compute and describe the flow field in the wakes. The science of using these NS equations to predict liquid and gas flows is called Computational Fluid Dynamics (CFD)[19]. However, due to their mathematical complexity, it is computationally expensive to use this to model the wake[24]. Simpler models to predict wake effects were developed. The first wake model was developed by Jensen (1983)[20], which was further developed by Jensen, Katic and Højstrup (1987)[21]. Jensen (1983) uses the assumption that the wake develops linearly with the downstream distance. Here the near wake region is neglected. This model is often called the Park model. Karakouzian et al. (2022) then applied this model to airborne wind energy systems (AWES)[23]. The Park model, together with the adaptation of Karakouzian et al. (2022), will be further discussed in section 4.1.

An entrainment-based[2] model published by Kaufman-Martin et al. (2022) tackles the wind farm design problem from another angle[49]. It uses the entrainment hypothesis to develop a lower-order model for the flow velocity of an annular wake generated by the flight path of an AWES. The entrainment hypothesis was introduced by G.I. Taylor et al. (1956) to model the impact of turbulent entrainment[51]. This hypothesis states that the mean inflow velocity across the boundary of a turbulent flow is proportional to a characteristic velocity of the flow [52]. This model is further discussed in 4.2.

Ainslie (1988) introduced another model in which the NS equation is replaced by an approximation of the thin shear layer and the viscous terms dropped[25]. In the paper, a wake model based on the numerical solution of the NS equations is given. For this an eddy viscosity (EV) turbulence model is used. Eddies are vortices whose flow direction differs from the general flow[26]. The formation of eddies is caused by a difference in pressure when the flow passes the wind turbine, causing it to swirl back on itself, becoming turbulent. The model relies on the difference between the free stream flow and the shear layer of the wake. This is a thin layer just behind the rotor with a concentrated vorticity and large variations in the tangential velocity[27]. Although this model is considered to

[1] The Navier-Stokes equations are partial differential equations, describing the relation between pressure, velocity, temperature and density in the flow field[18].

[2] Entrainment is a phenomenon in the atmosphere where non-turbulent flow is captured by turbulent flow

be reliable for conventional wind turbines, application to AWES goes hand in hand with assumptions for characteristic variables used in the model, which makes the EV use its credibility. Therefore the EV model is not further discussed in this research.

3 AWES

There are many different configurations of AWES, developed by various companies. However, all the designs of AWES have the same fundamental idea. It consists of an airborne device connected to a ground station by a tether[30]. This device then circles in the air in a figure-eight or circular pattern to gain velocity using crosswinds. Since a circular pattern gives us a higher cycle-averaged power output, this pattern will be used throughout the rest of this research[31].

The way of power generation for AWES can be split into two main concepts: onboard generation (drag-mode) and ground-based generation (lift-mode). Crosswind airborne power systems were first captured by Loyd (1980) in which lift- and drag-mode airborne power systems were first introduced[33]. As these systems rely on airborne devices heavier than air, both lift- and drag-mode rely on aerodynamic drag to be operational. Because the effect of wakes is investigated, it is assumed there is always enough aerodynamic drag for the airborne device to fly. In figure 1 these systems are visualized side by side.

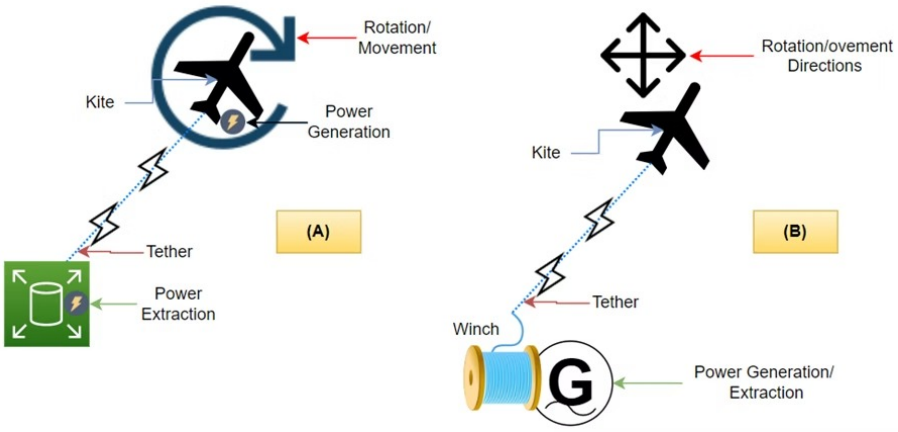


Figure 1: Visualization of (A) drag-mode and (B) lift-mode [32]

3.1 Lift- and drag-mode

The lift-mode airborne power systems make use of ground-based energy generation. An airborne device is directly attached to a winch, connected to an electric generator, by a tether. In lift-mode, usually, the airborne device is a kite. The kite uses crosswinds to fly in a helical pattern in the wind direction. This makes the tether reel out such that the winch rotates, driving the electric generator. This is known as the 'reel-out' phase. When the tether reaches maximum length, the kite changes pattern to a pattern producing much less lifting force. Then the tether is retracted by the winch, where the power consumption of the retraction phase is about 1% [36][38]. This cycle then repeats itself for continuous energy generation.

The drag-mode airborne power system uses on-board generation. Usually the airborne device used in drag-mode is a plane. Similar to the lift-mode system, the plane is connected by a tether to the ground station. However, here the electrical generators are on board the plane in the form of small turbines. These turbines generate power which is then transmitted through the tether to the ground station. Here the tether has a fixed length when in full operation, as opposed to the lift-mode system. An advantage of drag-mode systems is that the turbines can operate at a very high rotation speed in contrast to the slower-turning generators of HAWTs or lift-mode AWES[36].

Although both modes have the same fundamentals, they do have different characteristics. For example: Betz's law describes the maximum power that a wind turbine can extract from the wind, regardless of its aerodynamic composition. It is derived by taking the maximum of power coefficient C_p given by:

$$C_p = 4a(1-a)^2 \tag{1}$$

where induction factor a is the relation between the wind velocity at the rotor and the free-stream wind velocity[44].

$$a = \frac{u_\infty - u_r}{u_\infty} \tag{2}$$

Note that this is true for HAWTs and not necessarily for all AWES. For a HAWT this limit is set at 16/27 of the total wind power. For drag-mode AWES this limit is the same. In lift-mode AWES, however, this changes. The power coefficient of lift-mode AWES is dependent on a so-called reel-out factor, changing its equation. The maximum for the Betz limit in lift-mode is given by 4/27 of the wind power during the reel-out phase [39]. For the full derivation of these limits, visit [39].

Furthermore, because the plane in drag-mode systems takes on the same flight pattern, its wake is more consistent than the lift-mode system. Therefore it is easier to compute the effect its wake has on wind farms. With that being said, due to its mounted turbines, the drag coefficient C_D of the drag-mode AWES is 1.5 times as high as that of the lift-mode AWES. A higher drag coefficient results in a lower power generation, discussed in section 5. Drag-mode AWES will be shortened to 'AWES' for the remainder of this paper as this research focuses solely on the drag-mode systems.

3.2 Feasibility

A question that arises could be: Why would one prefer AWES over HAWTs? People are mainly interested in AWES for power generation because of the following reasons:

- In contrast to the wind turbines that are now operational in wind farms, AWES could capture the wind energy on higher altitudes[34]. Here the wind speed is typically higher and more consistent than closer to the sea level[35].
- Secondly, and most importantly, AWES can potentially have a much higher power-to-mass ratio than conventional renewable sources[36]. Because of this, large-scale

deployment of AWES is possible at a comparably low cost. Diehl et al. (2013) even state that theoretically, a system made out of HAWTs would weigh 300 times more than an airborne wind energy system for the same energy output.

- Lastly, because of the slim design and high altitude, it doesn't contribute to horizon pollution and has minimal to no ecological impact[37]. This could raise public acceptance of airborne wind farms.

However, as stated in Diehl et al. (2013), AWES do come with a price. To be operational, AWES need to fly. When a HAWT malfunctions, the operation can be shut down and repaired without any danger of damage. However, in the airborne case, a malfunction in the airborne wind system could cause the aerial device to crash, potentially destroying the system. The AWES need to be provided with sophisticated automatic control to prevent this. In this research the assumption is made that it is perfect.

4 The model

To find the energy production of AWES in a wind farm, first, a concept of an airborne device must be found. In this research, the airborne device used to compute our wake models is the Makani M600[60]. It is a drag-mode aeroplane with 8 turbines, designed by Makani Technologies LLC. It was designed for offshore conditions, making it viable for this research. The relevant parameters of the M600 are displayed in the table below.

Parameter	Units	Description	Intended	As-built
l	m	tether length	400	440
d_t	m	tether diameter	0.025	0.0295
A_{wing}	m^2	wing area	32.9	32.9
b	m	wing span	25.7	25.7
D	m	outer diameter of flightpath	75	145
d	m	inner diameter of flightpath	49.3	119.3
h_{min}	m	minimal altitude	85	110
m_{plane}	kg	plane mass	1310	1690
m_{tether}	kg	tether mass	315	390

Table 1: Parameters Makani M600

Figure 2: The Makani M600

4.1 Park model

To develop a wind farm for AWES the models mentioned in section 2 are reviewed. However not every model is viable for AWES. The wind farm wake model developed by Jensen (1983) can nevertheless be adapted to fit AWES. The model assumes a linear wake. Here the wake diameter (D_w) is dependent on the downstream distance x and rotor diameter (D) given by[21]:

$$D_w = D + \alpha x \tag{3}$$

α here is, following Jensen, called an entrainment constant. Here, entrainment refers to drawing the surrounding air into the wake[22]. This process helps the wake grow and dissipate over distance. The constant α here is given by [40]:

$$\alpha = \frac{1}{2ln\left(\frac{z}{z_0}\right)} \tag{4}$$

Where z is the hub height and z_0 is the surface roughness of the sea. Usually, the surface roughness is dependent on the site. However, for this research, the surface roughness is assumed to be constant. For offshore wind turbine sites, typically $z_0 = 0.0002$[41].

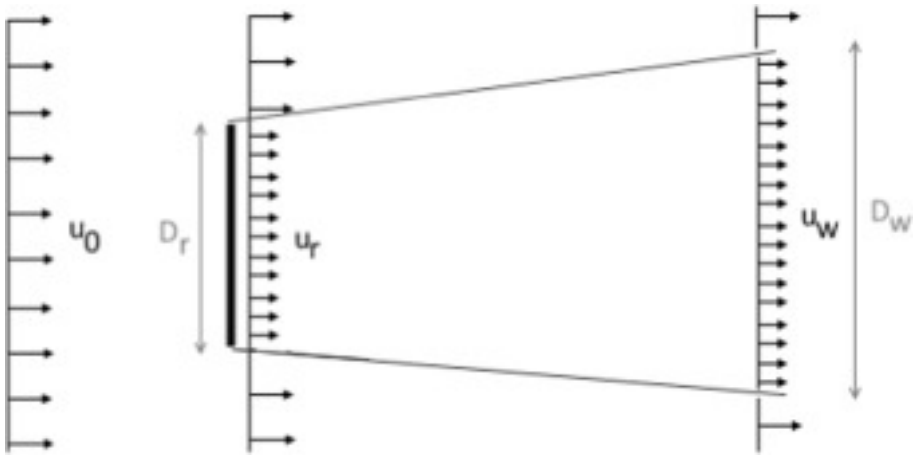


Figure 3: Visualization of Park wake model [42]

Wind velocity in the wake, u_w, is computed through the following equation:

$$u_w = (1 - \delta u)u_\infty \tag{5}$$

Where u_∞ is the velocity of the free-stream wind and δu is the velocity deficit at the turbine rotor given by:

$$\delta u = \frac{u_\infty - u_w}{u_\infty} = 1 - \frac{u_w}{u_\infty} \tag{6}$$

Since it is assumed in the Park model that the wake develops only in the stream direction, the velocity deficit as defined in equation 6 is only dependent on downstream distance x and no other dimensions.

To find δu, an expression which describes a relation between the different wind velocities must be found. Through the assumption of conservation of momentum and incompressibility[3] of the flow inside the wake, an equation satisfying this can be derived.

$$\begin{aligned} \sum density \cdot area \cdot velocity &= 0 \\ -\rho\pi\left(\frac{D}{2}\right)^2 u_r - \rho\pi\left(\left(\frac{D_w}{2}\right)^2 - \left(\frac{D}{2}\right)^2\right) u_\infty + \rho\pi\left(\frac{D_w}{2}\right)^2 u_w &= 0 \\ D^2 u_r + \left(D_w^2 - D^2\right) u_\infty &= D_w^2 u_w \end{aligned} \tag{7}$$

This however only gives an image of a wake in HAWT energy systems. To compute the wake effects of AWES using this same model, a couple of changes must be considered. In the figures 4 and 5 below, an image of the wake of AWES is given.

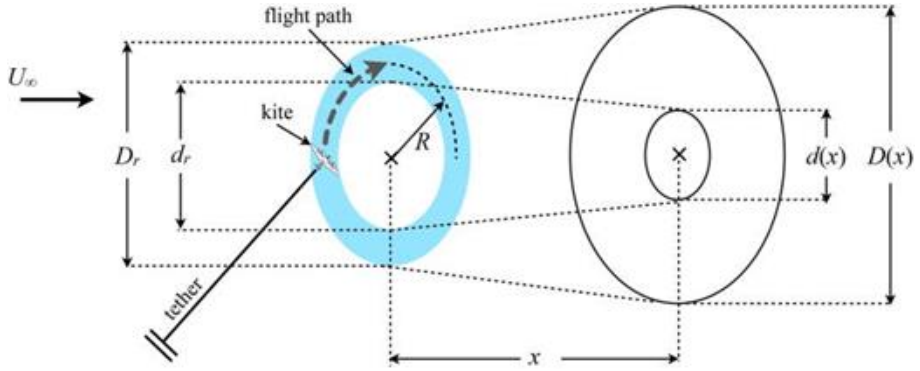


Figure 4: General overview of wake effect AWES [46]

Here the flight path of the aerial device (here a kite) is depicted, leaving a wake in the downstream direction. The outer diameter of the wake develops outwards, while the inner diameter develops inwards. This is then elaborated further schematically in figure[4] 5 below.

[3] A flow is considered incompressible whenever the density of the fluid in the flow remains constant[43]

[4] In figure 5 u_∞, u_r and u_w are denoted as U_∞, U_0 and $U(x)$ respectively.

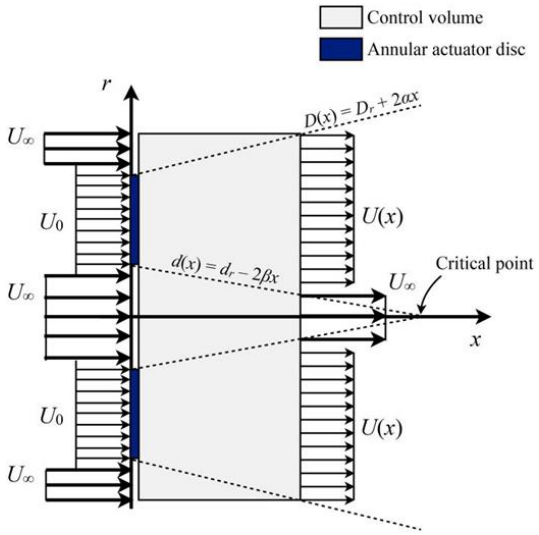


Figure 5: Visualization of Park wake model using AWES[46],
Here $d(x) = d_r - 2\beta x$ should be $d(x) = d_r - 2\alpha x$

To rewrite equation 7 such that it satisfies AWES wake effects, the equation must be adapted. With D being the outside diameter of the flight path, the inside diameter of the flight path is introduced as d. Using the entrainment constant α and integration factor $i(x)$ the inside diameter of the wake is given by[46]:

$$d_w = i(x)(d - 2\alpha x) \tag{8}$$

Where the integration factor $i(x)$ is given by

$$i(x) = \begin{cases} \frac{1}{2}\left(1 + \frac{d - 2\alpha x}{|d - 2\alpha x|}\right), & d - 2\alpha x \neq 0 \\ 0, & d - 2\alpha x = 0 \end{cases} \tag{9}$$

Since for $d - 2\alpha x > 0$, $i(x) = 1$ and for $d - 2\alpha x \leq 0$, $i(x) = 0$ the integration factor $i(x)$ ensures that d_w eventually goes to zero. Then the wake is no longer an annulus, but a circle.

In figure 5 the inside diameter of the wake is given by $d_w = i(x)(d - 2\beta x)$. Because the flight path is assumed to be axisymmetric and the airborne device to be rigid, $d_w = i(x)(d - 2\alpha x)$. Again, through the assumption of conservation of momentum and incompressibility:

$$
\begin{aligned}
\sum density \cdot area \cdot velocity &= 0 \\
-\rho\pi\left(\frac{d}{2}\right)^2 u_\infty - \rho\pi\left(\left(\frac{D_w}{2}\right)^2 - \left(\frac{D}{2}\right)^2\right) u_\infty - \rho\pi\left(\left(\frac{D}{2}\right)^2 - \left(\frac{d}{2}\right)^2\right) u_r & \\
+\rho\pi\left(\frac{d_w}{2}\right)^2 u_\infty + \rho\pi\left(\left(\frac{D_w}{2}\right)^2 - \left(\frac{d_w}{2}\right)^2\right) u_w &= 0 \\
\left(D^2 - d^2\right) u_r + \left(D_w^2 - D^2 + d^2 - d_w^2\right) u_\infty &= \left(D_w^2 - d_w^2\right) u_w
\end{aligned} \tag{10}
$$

Setting the initial velocity deficit right behind the flightpath in the axial direction as $\delta u = 1 - u_r/u_\infty$ and solving for u_w/u_∞ provides the following equation:

$$
\frac{u_w}{u_\infty} = 1 - \delta u_r \frac{D^2 - d^2}{D_w^2 - d_w^2} \tag{11}
$$

According to 1D momentum theory, at the end of the near-wake region[44]:

$$
u_w = (1 - 2a)u_\infty, \quad 0 \leq a \leq \frac{1}{2} \tag{12}
$$

where a is the induction factor[46]. The induction factor a is given in terms of the thrust coefficient C_T by[47]:

$$
a = 1 - \sqrt{1 - C_T} \tag{13}
$$

With the thrust coefficient given in terms of the thrust force T by:

$$
C_T = \frac{2T}{\rho A u^2} \tag{14}
$$

Although C_T depends on wind velocity u, the thrust coefficient does not increase with lower velocities. Since the thrust force T is equally and positively dependent on u, they cancel each other out. C_T is therefore considered to be constant.

Substituting equation 12 in 10 gives an expression for $\frac{u_w}{u_\infty}$:

$$
\frac{u_w}{u_\infty} = 1 - 2a \frac{D^2 - d^2}{D_w^2 - d_w^2} \tag{15}
$$

Combining equations 15 and 11 and substituting this and equation 13 in equation 6 provides the sought expression for the velocity deficit in the wake of an individual AWE system. This is given by:

$$
\begin{aligned}
\delta u(x) &= \frac{2\left(1 - \sqrt{1 - C_T}\right)\left(D^2 - d^2\right)}{D_w^2 - d_w^2} \\
&= \frac{2\left(1 - \sqrt{1 - C_T}\right)\left(D^2 - d^2\right)}{\left(D + 2\alpha x\right)^2 - i(x)^2\left(d - 2\alpha x\right)^2}
\end{aligned} \tag{16}
$$

In the above equation, only systems in the free stream are considered. In a wind farm, other AWE systems can be influenced by a system's wake, while not fully submerged by the wake. This is illustrated in Figure 6 by a wind farm using HAWTs.

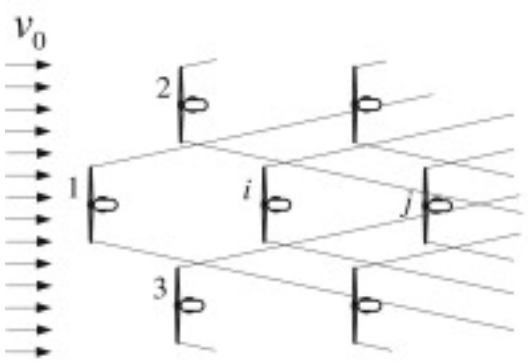


Figure 6: Wake effects in a HAWT wind farm[2]

To account for different systems, the initial variables need to be adjusted. Otherwise, the equations would become less orderly and more difficult to follow. Therefore, the wake-generating system is denoted as i and the system affected by the wake is denoted as j. Furthermore, a new variable is introduced: the incident wind speed u_{inc} given by

$$u_{inc} = (1 - \delta u)u_\infty \tag{17}$$

whenever the wind velocity is incident to a wind system in the free-stream. The incident wind velocity of wind system i is then denoted as $u_{inc,i}$. This is the wind speed at AWE system i before the wake is formed.

As previously mentioned, and seen in figure 6, wind systems can be influenced partially by another wind system upstream. To take this into consideration for the velocity deficit, a shadowing factor γ_α is introduced[48].

$$\gamma_\alpha = \frac{A_{intersection}}{A_{fligh path}}, \quad \gamma_\alpha \in [0, 1] \tag{18}$$

Here $A_{intersection}$ is the area of the intersection between the wake and the flight path. $A_{flightpath}$ is the area of the flight path. The equation for the velocity deficit is then changed as follows:

$$1 - \frac{u_w}{u_{inc,i}} = \frac{2\gamma_\alpha \left(1 - \sqrt{1 - C_T}\right) \left(D^2 - d^2\right)}{\left(D + 2\alpha x\right)^2 - i(x)^2 \left(d - 2\alpha x\right)^2} \tag{19}$$

This formulation now only formulates the wake effect on turbine i. To make the formulation more general the relation between the two turbines must be expressed. For i being a turbine upstream, and j being the turbine affected by i, equation 19 is generalized as follows:

$$\delta u_{ij} = 1 - \frac{u_{inc,j}}{u_{inc,i}} = \frac{2\gamma_\alpha \left(1 - \sqrt{1 - C_T}\right) \left(D^2 - d^2\right)}{\left(D + 2\alpha x\right)^2 - i(x)^2 \left(d - 2\alpha x\right)^2} \tag{20}$$

Whenever more wakes interact with each other, the velocity deficit is computed as follows:

$$\delta u_j = \sqrt{\sum_{i=1}^{N} \delta u_{ij}^2} \tag{21}$$

Here N is the number of interacting wakes and δu_{ij} is the velocity deficit of the wakes generated by system i affecting system j.

4.2 Kaufman-martin model

An entrainment-based model was developed by Kaufman-Martin et al.(2022)[49]. Here the assumption is made that the wake effects can be fully described by x-dependent u_w, d_w and D_w. The model uses the entrainment hypothesis to model the effects of turbulent entrainment. In the case of wake effects for the AWES, the flow downstream of the airborne device is assumed to be pulled into the wake, where the radial velocity is proportional to the downstream wind velocity. The entrainment from the external flow is given with radial velocity $w_e = E\,(u_\infty - u_w)$ and the entrainment from the core region by radial velocity $w_i = E\,(u_i - u_w)$, where u_i is the wind velocity in the core region, and E is an entrainment constant. Here E is an empirical, non-dimensional parameter, different to α from section 4.1[50].

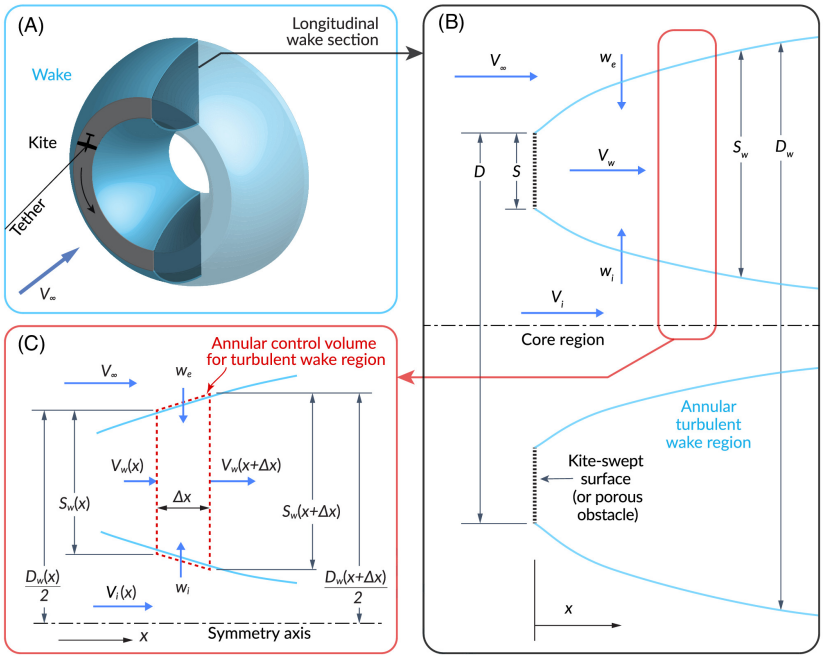


Figure 7: (A): Schematic view of the annular wake behind the airborne device. (B): Schematic cross-section of the annular wake. (C): Top part of the cross-section.

The model makes use of the conservation of momentum for the annular wake. It follows the approach comparable to that of previous studies on circular wakes [53]. Assuming the flow is incompressible, the conservation of mass in the annular control volume implies that

$$\begin{gathered}\frac{\rho\pi}{4}\left(D_w^2-d_w^2\right)u_w+\rho\pi D_w w_e\Delta x+\rho\pi\left(D_w-d_w\right)w_i\Delta x\\=\\\frac{\rho\pi}{4}\left(D_w^2-d_w^2\right)u_w+\frac{d}{dx}\left[\frac{\rho\pi}{4}\left(D_w^2-d_w^2\right)u_w\right]\Delta x\end{gathered}\tag{22}$$

Where the inflow mass flux[5] is given by the annular inflow $\frac{\rho\pi}{4}\left(D_w^2-d_w^2\right)u_w$ and the entrainment inflow $\rho\pi D_w w_e\Delta x+\rho\pi\left(D_w-d_w\right)w_i\Delta x$. The outflow leaving the annular region is given by $\frac{\rho\pi}{4}\left(D_w^2-d_w^2\right)u_w$ as well. $\frac{d}{dx}\left[\frac{\rho\pi}{4}\left(D_w^2-d_w^2\right)u_w\right]\Delta x$ accounts for the change in mass flux along the axial direction due to variations in wake properties.

After dividing by ρ,π and Δx, equation 22 can be simplified to:

$$\begin{aligned}\frac{d}{dx}\left[\frac{1}{4}\left(D_w^2-d_w^2\right)u_w\right] &= D_w w_e+\left(D_w-d_w\right)w_i\\ &= E\left(u_\infty-u_w\right)D_w+E\left(u_i-u_w\right)\left(D_w-d_w\right)\end{aligned}\tag{23}$$

This approach can then be used for the derivation of an equation for the conservation of momentum[6] in the annulus. This is based on the momentum coming into the control volume being the same as the momentum going out of the control volume. The terms in the conservation of mass equation 22 are then multiplied by the corresponding axial velocities. After simplifying, the following equation is obtained.

$$\frac{d}{dx}\left[\frac{1}{4}\left(D_w^2-d_w^2\right)u_w^2\right] = Eu_\infty\left(u_\infty-u_w\right)D_w+Eu_i\left(u_i-u_w\right)\left(D_w-d_w\right)\tag{24}$$

Using the same approach it is possible to find the equations for the conservation of mass and momentum in the core region of the flow. The simplified version of these equations is given in equations 25.

$$\begin{aligned}\frac{d}{dx}\left[\frac{1}{4}d_w^2u_i\right] &= -E\left(u_i-u_w\right)d_w\\ \frac{d}{dx}\left[\frac{1}{4}d_w^2u_i^2\right] &= -Eu_i\left(u_i-u_w\right)d_w\end{aligned}\tag{25}$$

Performing the product rule on the equation below from equations 25 gives:

$$\frac{d}{dx}\left[\frac{1}{4}d_w^2u_i\right]u_i+\frac{1}{4}d_w^2u_i\frac{du_i}{dx} = -Eu_i\left(u_i-u_w\right)d_w\tag{26}$$

Multiplying equation 25 by u_i and subtracting it from equation 26 then gives

$$\frac{1}{4}d_w^2u_i\frac{du_i}{dx}=0\tag{27}$$

[5]Mass flux is defined as the amount of mass transported across a unit area perpendicular to the direction of mass transport, per unit time[54]

[6]Momentum is defined to be the product of the mass of a particle and its velocity[55]

which implies

$$u_i = \text{constant} \tag{28}$$

Using one-dimensional momentum theory, the assumption is made that $u_i = u_\infty$ as there is no interference in the core region. Then the equations 23, 24 and 25 can be rewritten as follows:

$$\begin{aligned}
\frac{d}{dx}\left[\frac{1}{4}\left(D_w^2 - d_w^2\right)u_w\right] &= E\left(u_\infty - u_w\right)\left(2D_w - d_w\right) \\
\frac{d}{dx}\left[\frac{1}{4}\left(D_w^2 - d_w^2\right)u_w^2\right] &= Eu_\infty\left(u_\infty - u_w\right)\left(2D_w - d_w\right) \\
\frac{d}{dx}\left[\frac{1}{4}d_w^2 u_\infty\right] &= -E\left(u_\infty - u_w\right)d_w
\end{aligned} \tag{29}$$

Solving this set of ODEs seems above the level of the Bachelor of Applied Mathematics; however, after rewriting this and introducing new variables it is possible to solve this set of ODEs numerically. In the following expressions, M is the momentum flux and m denotes the mass flux. m_i is the mass flux in the core region. Here ρ and π are factored out as well

$$\begin{aligned}
m_w &= \frac{1}{4}\left(D_w^2 - d_w^2\right)u_w \\
M_w &= \frac{1}{4}\left(D_w^2 - d_w^2\right)u_w^2 \\
m_i &= \frac{1}{4}d_w^2 u_\infty
\end{aligned} \tag{30}$$

These equations are then used to rewrite the system of ODEs in 29 to the following system:

$$\begin{aligned}
\frac{dm_w}{dx} &= 2E\left(u_\infty - \frac{M_w}{m_w}\right)\left(\sqrt{\frac{m_i}{u_\infty} + \frac{m_w^2}{M_w}} + \sqrt{\frac{m_i}{u_\infty}}\right) \\
\frac{dM_w}{dx} &= u_\infty \frac{dm_w}{dx} \\
\frac{dm_i}{dx} &= -2E\left(u_\infty - \frac{M_w}{m_w}\right)\sqrt{\frac{m_i}{u_\infty}}
\end{aligned} \tag{31}$$

After computing the results of the above system, the results are converted back into its desired form by the following equations:

$$\begin{aligned}
u_w &= \frac{M_w}{m_w} \\
d_w &= 2\sqrt{\frac{m_i}{u_\infty}} \\
D_w &= 2\sqrt{\frac{m_i}{u_\infty} + \frac{m_w^2}{M_w}}
\end{aligned} \tag{32}$$

The same input parameters for this model were used as in Haas & Meyers (2017)[57]. For the laminar inflow case entrainment constant $E = 0.15$ is chosen. For the turbulent inflow case $E = 0.5$. Also, the induction factor a was taken from this model and denoted as $a = 1/3$, which is the Betz limit for drag-mode AWES. The initial conditions derived by Kaufman-Martin et al. (2021) are given by the following equations, after which $a = 1/3$ is substituted [49]:

$$\begin{aligned} u_{w,0} &= u_\infty(1-2a) &&= \frac{1}{3}u_\infty \\ D_{w,0} &= \sqrt{D^2 + \frac{1}{4}(D^2-d^2)\frac{4a}{1-2a}} &&= \sqrt{2D^2-d^2} \\ d_{w,0} &= d \end{aligned} \tag{33}$$

4.2.1 Numerical approximation

Various methods were considered to solve the system portrayed by equations 31 numerically. All methods are based on time integration based on t. In this system an approximation of the ODE is obtained using x as our 'time' variable. This means that instead of time, space is used as a variable. Only since wake effects are x-dependent, instead of time dependent, this is possible.

Firstly, the Forward Euler method was considered. This is the simplest and best-known time-integration method. In the general case the numerical approximation of t_{n+1} denoted by[45]

$$w_{n+1} = w_n + \Delta t f(t_n, w_n) \tag{34}$$

Where $f(t_n, w_n)$ is the derivative of w_n with respect to t_n. Since w_{n+1} can be computed directly from 34 it is called an explicit method. Despite that this method has a low computational intensity, it can't be used to solve the system 31 numerically. The Forward Euler method does not always provide a stable solution to a complex problem[56]. Therefore this method was not chosen as a method to solve the system 31.

An unconditionally stable method is given by the Backward Euler method. The numerical approximation at time t_{n+1} using the Backward Euler method is expressed by

$$w_{n+1} = w_n + \Delta t f(t_{n+1}, w_{n+1}) \tag{35}$$

Note that the term w_{n+1} is included in both sides of the equations. For that reason the Backward Euler method is called an implicit method. If f were to be linearly dependent on w then the solution to 35 is easily computed. For nonlinear initial-value problems however, the solution to 35 is nontrivial[45]. Numerical nonlinear solvers such as the Newton-Raphson method need to be implemented. This implies that the computational intensity at every time step of the Backward Euler method is high. To obtain accurate results the computational cost of the Backward Euler method would be high when this method would be used to solve system 31.

Finally, the fourth-order method of Runge-Kutta is considered. Despite its explicit nature, the Runge-Kutta method (RK4 method) has attractive stability properties. The RK4 method uses predictors k_1, k_2, k_3 and k_4 to approximate the solution at t_{n+1} by

$$w_{n+1} = w_n + \frac{1}{6}(k_1 + 2k_2 + 2k_3 + k_4) \tag{36}$$

The predictors are denoted by:

$$\begin{aligned} k_1 &= \Delta t f(t_n, w_n) \\ k_2 &= \Delta t f\left(t_n + \frac{1}{2}\Delta t, w_n + \frac{1}{2}k_1\right) \\ k_3 &= \Delta t f\left(t_n + \frac{1}{2}\Delta t, w_n + \frac{1}{2}k_2\right) \\ k_4 &= \Delta t f\left(t_n + \Delta t, w_n + k_3\right) \end{aligned} \tag{37}$$

To apply the RK4 method to the system of equations in 31 the numerical approximation denoted in equation 36 must be transformed. For m_w the solution at x_{n+1} is approximated by

$$m_{w,n+1} = m_{w,n} + \frac{1}{6}\left(k_1 + 2k_2 + 2k_3 + k_4\right) \tag{38}$$

With k_1, k_2, k_3 and k_4 again the predictors. Using that

$$f(x_n, m_{w,n}) = 2E\left(u_\infty - \frac{M_w}{m_w}\right)\left(\sqrt{\frac{m_i}{u_\infty} + \frac{m_w^2}{M_w}} + \sqrt{\frac{m_i}{u_\infty}}\right) \tag{39}$$

extracted from equations 31. The now Δx-dependent predictors are denoted by

$$\begin{aligned} k_1 &= \Delta x f(x_n, m_{w,n}) \\ k_2 &= \Delta x f\left(x_n + \frac{1}{2}\Delta x, m_{w,n} + \frac{1}{2}k_1\right) \\ k_3 &= \Delta x f\left(x_n + \frac{1}{2}\Delta x, m_{w,n} + \frac{1}{2}k_2\right) \\ k_4 &= \Delta x f\left(x_n + \Delta x, m_{w,n} + k_3\right) \end{aligned} \tag{40}$$

Similarly the numerical approximations are found for M_w and m_i at x_{n+1}. For M_w the solution at x_{n+1} is approximated by

$$M_{w,n+1} = M_{w,n} + \frac{1}{6}\left(k_1 + 2k_2 + 2k_3 + k_4\right) \tag{41}$$

With the predictors now given by

$$\begin{aligned} k_1 &= \Delta x g(x_n, M_{w,n}) \\ k_2 &= \Delta x g\left(x_n + \frac{1}{2}\Delta x, M_{w,n} + \frac{1}{2}k_1\right) \\ k_3 &= \Delta x g\left(x_n + \frac{1}{2}\Delta x, M_{w,n} + \frac{1}{2}k_2\right) \\ k_4 &= \Delta x g\left(x_n + \Delta x, M_{w,n} + k_3\right) \end{aligned} \tag{42}$$

With $g(x_n, M_{w,n})$ given by:

$$g(x_n, M_{w,n}) = 2E\left(u_\infty - \frac{M_w}{m_w}\right)\left(\sqrt{\frac{m_i}{u_\infty} + \frac{m_w^2}{M_w}} + \sqrt{\frac{m_i}{u_\infty}}\right)u_\infty \tag{43}$$

Derived from equations 31. Lastly the numerical approximation for m_i at x_{n+1} is found by

$$m_{i,n+1} = m_{i,n} + \frac{1}{6}\left(k_1 + 2k_2 + 2k_3 + k_4\right) \tag{44}$$

using predictors:

$$\begin{aligned} k_1 &= \Delta x h(x_n, m_{i,n}) \\ k_2 &= \Delta x h\left(x_n + \frac{1}{2}\Delta x, m_{i,n} + \frac{1}{2}k_1\right) \\ k_3 &= \Delta x h\left(x_n + \frac{1}{2}\Delta x, m_{i,n} + \frac{1}{2}k_2\right) \\ k_4 &= \Delta x h\left(x_n + \Delta x, m_{i,n} + k_3\right) \end{aligned} \tag{45}$$

With the function $h(x_n, m_{i,n})$ given by the equation below

$$h(x_n, m_{i,n}) = -2E\left(u_\infty - \frac{M_w}{m_w}\right)\sqrt{\frac{m_i}{u_\infty}} \tag{46}$$

again from equations 31.

4.2.2 Stability

For the general form applying $\frac{dy}{dt} = \lambda y$ to the system, where y is the analytical solution of the problem, results in

$$w_{n+1} = \left(1 + \lambda\Delta t + \frac{1}{2}(\lambda\Delta t)^2 + \frac{1}{6}(\lambda\Delta t)^3 + \frac{1}{24}(\lambda\Delta t)^4\right)w_n \tag{47}$$

Therefore the amplification factor is expressed as

$$Q(\lambda\Delta t) = 1 + \lambda\Delta t + \frac{1}{2}(\lambda\Delta t)^2 + \frac{1}{6}(\lambda\Delta t)^3 + \frac{1}{24}(\lambda\Delta t)^4 \tag{48}$$

Note that the numerical scheme is stable if and only if

$$|Q(\lambda\Delta t)| \leq 1 \tag{49}$$

Whenever $\lambda \geq 0$ the amplification factor $Q(\lambda\Delta t)$ is bigger than one, hence not stable. Now for $\lambda < 0$ the RK4 method is stable whenever

$$-1 \leq 1 + \lambda\Delta t + \frac{1}{2}(\lambda\Delta t)^2 + \frac{1}{6}(\lambda\Delta t)^3 + \frac{1}{24}(\lambda\Delta t)^4 \leq 1 \tag{50}$$

To simplify the equation $\lambda\Delta t$ is denoted as a. For the LHS this results in the following inequality:

$$0 \leq 2 + a + \frac{1}{2}a^2 + \frac{1}{6}a^3 + \frac{1}{24}a^4 \tag{51}$$

This gives a polynomial given by

$$P(a) = 2 + a + \frac{1}{2}a^2 + \frac{1}{6}a^3 + \frac{1}{24}a^4 \tag{52}$$

The extrema of this polynomial lie at the zeroes of P', which is denoted by

$$P'(a) = 1 + a + \frac{1}{2}a^2 + \frac{1}{6}a^3 = 0 \tag{53}$$

Assume there is an arbitrary extreme value of the polynomial denoted by $\tilde{a}$. Then

$$\begin{aligned} P(\tilde{a}) &= 2 + \tilde{a} + \frac{1}{2}\tilde{a}^2 + \frac{1}{6}\tilde{a}^3 + \frac{1}{24}\tilde{a}^4 \\ &= 1 + P'(\tilde{a}) + \frac{1}{24}\tilde{a}^4 \\ &= 1 + \frac{1}{24}\tilde{a}^4 \end{aligned} \tag{54}$$

Since $1 + \frac{1}{24}\tilde{a}^4 > 0$ for all $\tilde{a}$. Therefore all extreme values are positive, which means that the minimum of the polynomial is positive as well. Subsequently, $P(a)$ is positive for all a. This leaves the inequality 50 with

$$1 + a + \frac{1}{2}a^2 + \frac{1}{6}a^3 + \frac{1}{24}a^4 \leq 1 \tag{55}$$

Which is then written as

$$a\left(1 + \frac{1}{2}a + \frac{1}{6}a^2 + \frac{1}{24}a^3\right) \leq 0 \tag{56}$$

Since $a = \lambda\Delta t < 0$, equation 56 is equivalent to

$$1 + \frac{1}{2}a + \frac{1}{6}a^2 + \frac{1}{24}a^3 \geq 0 \tag{57}$$

Setting the polynomial to 0 and then solving it for a gives only one zero which lies at $a \approx -2.8$. The polynomial is negative for $a < -2.8$ and positive for $a > -2.8$. Therefore $a = \lambda\Delta t > -2.8$ or

$$\Delta t < -\frac{2.8}{\lambda} \tag{58}$$

Then $|Q(\lambda\Delta t)| \leq 1$ is satisfied.

Setting up the matrix

Now it is possible to set up the matrix. Define state vector $\mathbf{y}(x)$ as follows:

$$\mathbf{y}(x) = \begin{bmatrix} m_w(x) \\ M_w(x) \\ m_i(x) \end{bmatrix} \tag{59}$$

With the system of equations as in 31 denoted by:

$$\begin{cases} \dfrac{dm_w}{dx} &= 2E\left(u_\infty - \dfrac{M_w}{m_w}\right)\left(\sqrt{\dfrac{m_i}{u_\infty} + \dfrac{m_w^2}{M_w}} + \sqrt{\dfrac{m_i}{u_\infty}}\right) \\ \dfrac{dM_w}{dx} &= 2E\left(u_\infty - \dfrac{M_w}{m_w}\right)\left(\sqrt{\dfrac{m_i}{u_\infty} + \dfrac{m_w^2}{M_w}} + \sqrt{\dfrac{m_i}{u_\infty}}\right)u_\infty \\ \dfrac{dm_i}{dx} &= -2E\left(u_\infty - \dfrac{M_w}{m_w}\right)\sqrt{\dfrac{m_i}{u_\infty}} \end{cases} \tag{60}$$

The stability is checked using the initial conditions of system 30. The initial conditions of 30 are based on the initial conditions $D_{w,0}, d_{w,0}$ and $u_{w,0}$ which are denoted in the system of equations 33 given below, using Betz limit $a = 1/3$:

$$\begin{aligned} u_{w,0} &= u_\infty(1-2a) &&= \frac{1}{3}u_\infty &&= 2.667\text{m/s} \\ D_{w,0} &= \sqrt{D^2 + \frac{1}{4}(D^2-d^2)\frac{4a}{1-2a}} &&= \sqrt{2D^2-d^2} &&= 166.8\text{m} \\ d_{w,0} &= d &&&&= 119.3\text{m} \end{aligned} \tag{61}$$

Plugging this into 30 gives:

$$\begin{aligned} m_{w,0} &= \frac{1}{4}\left(D_{w,0}^2 - d_{w,0}^2\right)u_{w,0} &&= 9059.8 \\ M_{w,0} &= \frac{1}{4}\left(D_{w,0}^2 - d_{w,0}^2\right)u_{w,0}^2 &&= 24160 \\ m_{i,0} &= \frac{1}{4}d_{w,0}^2 u_\infty &&= 28465 \end{aligned} \tag{62}$$

Now the Jacobian matrix of the system 60 can be computed: $J = \dfrac{\partial \mathbf{f}}{\partial \mathbf{y}}$ where $\mathbf{f} = [f, g, h]^T$ are the right hand sides of the system 60. The Jacobian is then computed:

$$J = \begin{bmatrix} \dfrac{\partial f}{\partial m_w} & \dfrac{\partial f}{\partial M_w} & \dfrac{\partial f}{\partial m_i} \\ \dfrac{\partial g}{\partial m_w} & \dfrac{\partial g}{\partial M_w} & \dfrac{\partial g}{\partial m_i} \\ \dfrac{\partial h}{\partial m_w} & \dfrac{\partial h}{\partial M_w} & \dfrac{\partial h}{\partial m_i} \end{bmatrix} \tag{63}$$

Substituting initial values 62 together with entrainment constant $E = 0.31$ and free stream wind velocity $u_\infty = 8.0$ the Jacobian takes on the following values:

$$J = \begin{bmatrix} 0.041 & -0.013 & 0.006 \\ 0.328 & -0.101 & 0.048 \\ -0.011 & 0.004 & -0.003 \end{bmatrix} \tag{64}$$

The eigenvalues derived from this Jacobian are $\lambda_1 = -0.060, \lambda_2 = -1.4 * 10^{-3} + 8.6i * 10^{-4}$ and $\lambda_3 = -1.4 * 10^{-3} - 8.6i * 10^{-4}$. The real parts of the eigenvalues are below zero, meaning that the system is stable. The stability is checked for stepsize using equation 58 as follows:

$$\Delta x < -\frac{2.8}{\lambda_{min}} = \frac{2.8}{0.060} = 46.667 \tag{65}$$

Hence the system is stable for a stepsize of $\Delta x < 46$ as for this λ and Δx the amplification factor $|Q(\lambda \Delta x)| \leq 1$. In the implementation of the Kaufman-Martin model, a stepsize of $\Delta x = 0.1$ meters is used.

5 Implementation

The performance of wind farms is measured using their annual energy production (AEP) or just their power generation. This will also be done for the AWES wind farms, which consist of the Makani M600 systems. The power equation used for HAWTs in conventional wind energy is given by:

$$P = \frac{1}{2}\rho C_p \pi R^2 u^3 \tag{66}$$

Where C_p is the thrust coefficient and R is the span of a wind turbine blade. As estimated by Loyd (1980), the power P that a drag AWES can generate under idealized assumptions can be approximated by[33][36]:

$$P = \frac{2}{27}\rho A_{wing} u^3 C_L \left(\frac{C_L}{C_D}\right)^2 \tag{67}$$

where C_L is the lift coefficient, C_D is the drag coefficient and A_{wing} is the area of the wing. The lift and drag coefficients are given by

$$C_D = \frac{2D}{\rho u^2 A_{wing}} \tag{68}$$

$$C_L = \frac{2L}{\rho u^2 A_{wing}} \tag{69}$$

Where D and L are the drag and lift forces, respectively. Again, since D and C_D are equally dependent on u in opposite directions, the drag coefficient C_D doesn't shrink. This holds for the lift coefficient C_L as well. Now, the power equation is only given in the most general form. The power generated by Makani M600 j is expressed as follows:

$$P_j = \frac{2}{27}\rho A_{wing} C_L \left(\frac{C_L}{C_D}\right)^2 u_{inc,j}^3 \tag{70}$$

C_D and C_L are entirely dependent on the specs of the Makani M600 which are displayed in table 1. First, the intended parameters of the Makani M600 were considered to be used. Here the drag and lift coefficients equalled 0.26 and 2.8 respectively. This would however mean that the power capacity of one Makani M600 calculated by 67 would only be 1.8 times less than that of the Vestas 90-3MW in contrary to reality. Since the M600 is designed for a power capacity of 600kW and the Vestas for 3MW, the power ratio between the M600 and the Vestas must be closer to 5. Therefore the parameters of the M600 as-built were taken for the implementation with $C_D = 0.312$ and $C_L = 2.56$. This version has a rated[7] power capacity of 685 kW at a rated wind speed of 11 m/s, which is closer to the real value. This version will therefore be used in the implementation as well.

[7] Rated power signifies the possible power output the appliance, here M600, can generate under ideal conditions[64]

5.1 OWEZ

For the first simulation, the Offshore Windpark Egmond aan Zee is recreated. However, instead of HAWTs, now the Makani M600 airborne energy systems are used. These Makani energy systems are located at the same locations as the 36 turbines of the OWEZ such that they can be compared. The wind turbines used in the OWEZ are of the Vestas 90-3MW model. The layout of the OWEZ is given in the figure below.

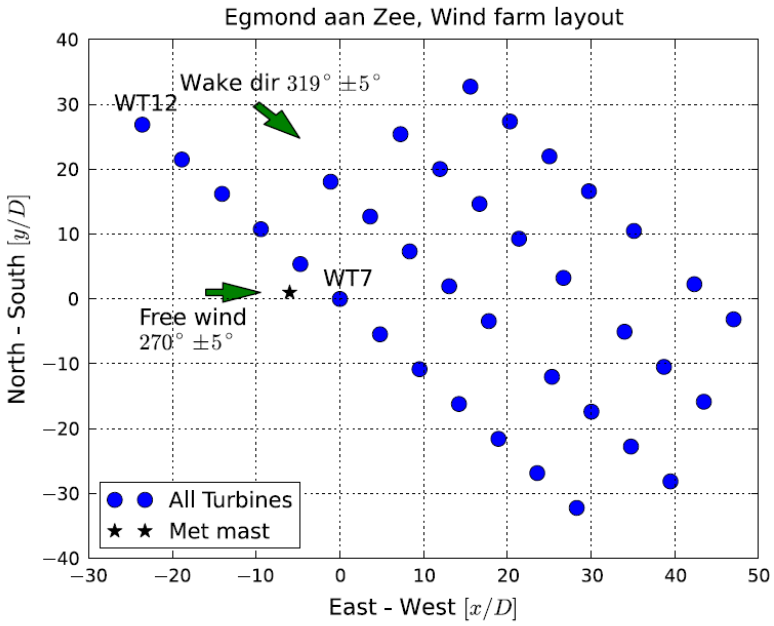


Figure 8: Layout of the OWEZ

Larsen et al. (2013) have analysed the OWEZ. Using a meteorological mast (met mast) they were able to do high-quality measurements on turbine loads. Despite the high fidelity of the met mast, wind speed and direction were difficult to measure because of the wake effects generated by the wind turbines. For this, they used the anemometer of wind turbine 12 (WT12) at a wind direction of 319°. At this angle the wake effects of the turbines are the highest, as here the turbines are aligned and the distance between them is the smallest. This could be a great example of measuring the wake effects of the Makani M600, which were first thought to be negligible. The figure below shows the farm's power production for every wind direction. Our point of interest will be the 319° wind direction.

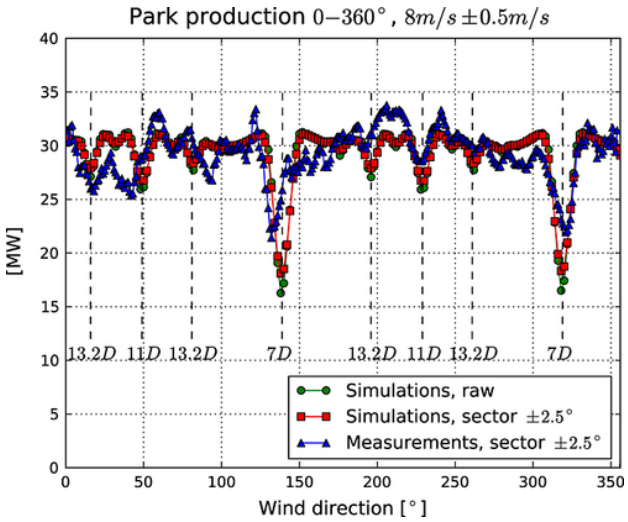


Figure 9: Energy production OWEZ, per wind direction[57]

In this first simulation the turbines were ordered. Each energy system was given a number, where system 0 was chosen as the origin of the farm system. Since the 4 rows of wind turbines are a distance of ≈700 meters apart, the wake effects between the rows are negligible. The M600's were numbered as follows:

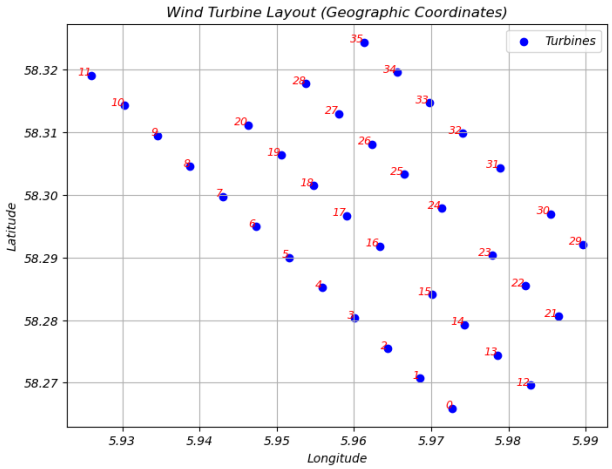
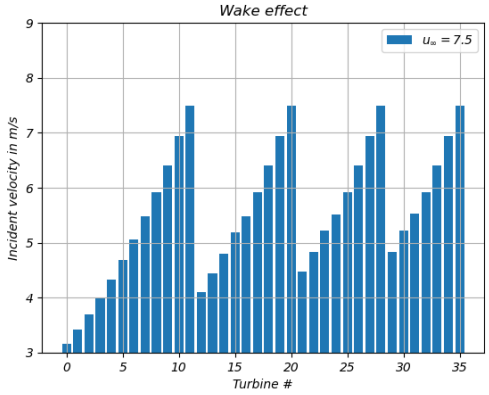


Figure 10: Schematic layout of OWEZ at its geographic coordinates[62]

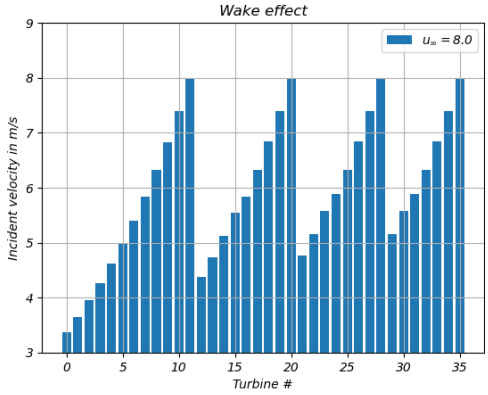
5.1.1 Park model

Now to implement the Park model described in 4.1. To achieve this, the thrust coefficient C_T is needed. As the assumption is made that the plane in our AWES flies at a constant speed, by Newton's first law the two opposite forces, thrust and drag, must be the same. Subsequently, $T = D \rightarrow C_T = C_D = 0.312$.

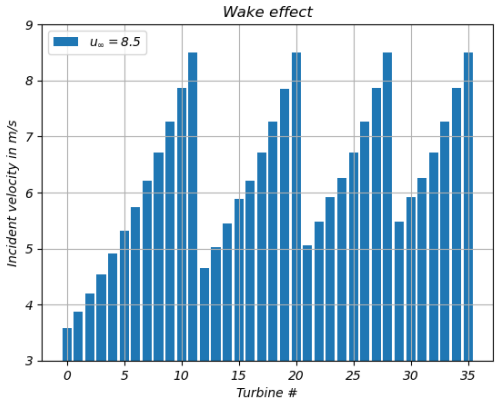
To compare the values found by the offshore wind farm made out of M600's the total power generation needs to be found for wind speeds between 7.5 m/s and 8.5 m/s such that the results can be compared to figure 9. The incident wind speed at each system was found using the method described in 4.1. They are displayed in the Figures below.



(a) Wake effect at $u_\infty = 7.5$



(b) Wake effect at $u_\infty = 8.0$



(c) Wake effect at $u_\infty = 8.5$

Figure 11: The incident velocities for $7.5 \leq u_\infty \leq 8.5$ using the Park model

These give a visual overview of the wake effects at different wind speeds. To give a more complete view of the wake effects the incident velocities for a free stream wind velocity

of 8.0 m/s are given in the tables below.

#M600	0	1	2	3	4	5	6	7	8	9	10	11
$u_\infty = 8.0,\ \ (\mathrm{m/s})$	3.37	3.64	3.95	4.27	4.62	5.00	5.40	5.84	6.32	6.83	7.40	8.00

Table 2: First row

#M600	12	13	14	15	16	17	18	19	20
$u_\infty = 8.0,\ \ (\mathrm{m/s})$	4.38	4.73	5.12	5.54	5.84	6.32	6.84	7.40	8.00

Table 3: Second row

#M600	21	22	23	24	25	26	27	28
$u_\infty = 8.0,\ \ (\mathrm{m/s})$	4.76	5.15	5.57	5.88	6.32	6.84	7.40	8.00

Table 4: Third row

#M600	29	30	31	32	33	34	35
$u_\infty = 8.0,\ \ (\mathrm{m/s})$	5.15	5.57	5.89	6.32	6.84	7.40	8.00

Table 5: Fourth row

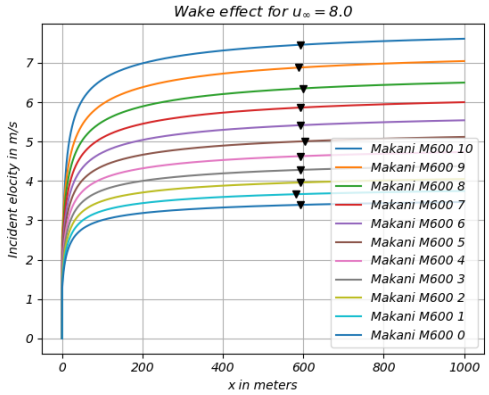
The total power generation for wind speeds between 7.5 m/s and 8.5 m/s is then calculated by taking the incident wind speed for each system and filling it in in equation 70. These were subsequently summed which gave a total power generation between approximately 3.64 MW and 5.30 MW. It must be noted that these values are reached for ideal circumstances. Comparing these results to figure 9 indicates that the power generation of conventional wind turbines is still higher than that of the M600's, which is in line with expectations. Using figure 9, the power generation at a wind velocity of $u_\infty = 8.0$ m/s is approximated to be 22MW for a conventional turbine for an angle of attack of 319°. Thereafter, it is quickly observed that the power generation of the M600 wind farm, as calculated, is 4-6 times smaller. The power capacity of the Makani M600 is 5 times smaller than that of the Vestas 90-3MW, hence the calculations are in line with literature [58][59].

Further comparison of the M600 wind farm and the OWEZ leads us to their difference in mass. Firstly, the individual AWE system of the Makani M600 is considered. In table 1 the mass and the tether are displayed. This gives a total of 2080kg for the airborne device itself. The major components of the ground station are the winch frame, perch, drum and tower, which weigh 6200 kg, 900 kg, 2500 kg and 9000 kg respectively[61]. Combine this with the Makani M600 result of 20 680 kg. The weight of the Vestas 90-3MW is made up of its rotor, nacelle and tower, which weigh 41,000 kg, 70,000 kg and 285,000 kg respectively. Summed up, this results in a total weight of 396,000 kg.

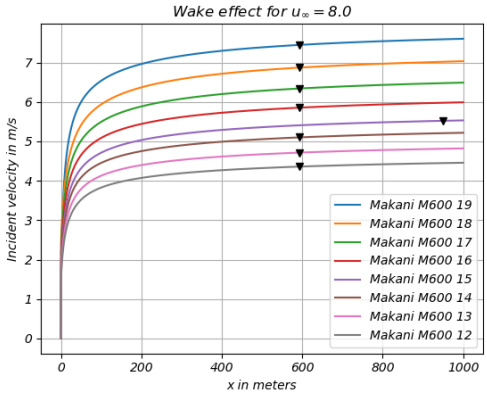
Combining and then comparing the power and mass of both systems make for an interesting new characteristic. The power-to-mass ratio is an effective term to compare both the Makani M600 and the Vestas 90-3MW. When the M600 and Vestas reach their rated capacity of 600kW and 3MW respectively, their power-to-mass ratio equals 29.0W/kg and 7.58W/kg. This is a difference in power-to-mass ratio of a factor of 3.8. What would these values become when these systems are situated in the OWEZ? Before answering this question, first, the Kaufman-Martin model elaborated in 4.2 will be discussed.

5.1.2 Kaufman-Martin model

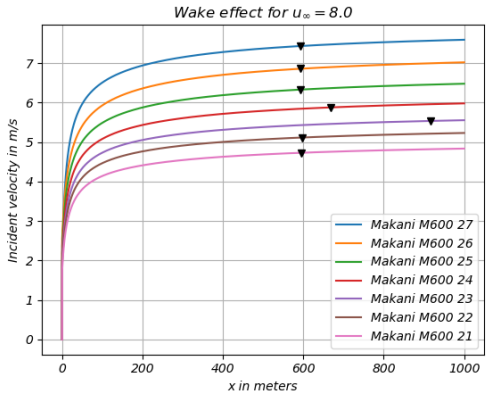
Now, the Kaufman-Martin is considered for the OWEZ. Again the Makani M600 systems are in the exact location as the wind turbines of the OWEZ. Following the numerical approximation described in 4.2.1 results were found for the wake effects incident to the M600s. The entrainment constant E was set at 0.31. Using a time step of $\Delta x = 0.1$, a total number of steps of 10000 and a free stream velocity of $u_\infty = 8.0$, the figures below were obtained:



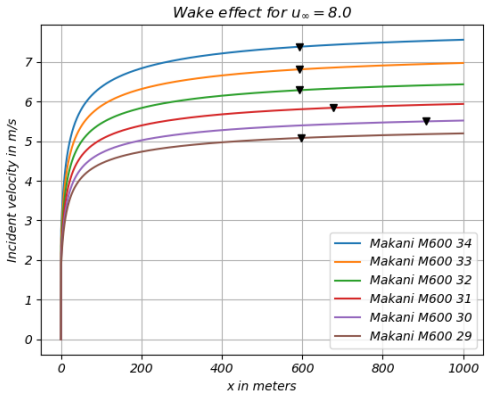
(a) Wake effect at row 1



(b) Wake effect at row 2



(c) Wake effect at row 3



(d) Wake effect at row 4

Figure 12: Incident velocities for $u_\infty = 8.0$

In the figures the incident velocity of the Makani M600 at place 11, 20, 28 and 35 are not shown as for these systems the incident velocity equals the free stream velocity u_∞. The plots indicate what the incident velocity would be for a distance of x between the system downstream. The black triangles indicate at what distance the M600 is to the M600 system downstream whereafter this velocity is taken as the incident velocity for that particular system. For example, in graph 14b all M600s are at a distance of ≈600m within each other except M600 #15. This M600 is at a distance of ≈950m downstream of M600 #16.

In the tables below again the incident velocities at the M600's are displayed:

#M600	0	1	2	3	4	5	6	7	8	9	10	11
$u_\infty = 8.0,\ \text{(m/s)}$	3.39	3.66	3.96	4.28	4.63	5.01	5.42	5.86	6.35	6.88	7.46	8.00

Table 6: First row

#M600	12	13	14	15	16	17	18	19	20
$u_\infty = 8.0,\ \text{(m/s)}$	4.36	4.71	5.10	5.52	5.85	6.34	6.87	7.45	8.00

Table 7: Second row

#M600	21	22	23	24	25	26	27	28
$u_\infty = 8.0,\ \text{(m/s)}$	4.73	5.12	5.54	5.88	6.33	6.86	7.44	8.00

Table 8: Third row

#M600	29	30	31	32	33	34	35
$u_\infty = 8.0,\ \text{(m/s)}$	5.08	5.50	5.84	6.28	6.81	7.38	8.00

Table 9: Fourth row

The total power output is again computed for the free stream velocities $u_\infty = 7.5$ m/s and $u_\infty = 8.5$ m/s. Between these velocities, the total power generation of the M600 wind farm is calculated to be between approximately 3.65MW and 5.31MW. Here again, the power-to-mass ratio of the Makani M600 wind farm is 4 - 6 times larger than that of the OWEZ, using the values in figure 9. In the next section the Park model and the Kaufman-Martin model are compared.

5.1.3 Comparison

For both the Park and Kaufman-Martin models were evaluated using the OWEZ wind turbine locations. The incident wind speeds here were computed at a free stream wind velocity of $u_\infty = 8.0$ m/s. These values are put together in the tables below for comparison.

#M600	0	1	2	3	4	5	6	7	8	9	10	11
Park, (m/s)	3.37	3.64	3.95	4.27	4.62	5.00	5.40	5.84	6.32	6.83	7.40	8.00
KM, (m/s)	3.39	3.66	3.96	4.28	4.63	5.01	5.42	5.86	6.35	6.88	7.46	8.00

Table 10: First row, $u_\infty = 8.0$ m/s

#M600	12	13	14	15	16	17	18	19	20
Park, (m/s)	4.38	4.73	5.12	5.54	5.84	6.32	6.84	7.40	8.00
KM, (m/s)	4.36	4.71	5.10	5.52	5.85	6.34	6.87	7.45	8.00

Table 11: Second row, $u_\infty = 8.0$ m/s

#M600	21	22	23	24	25	26	27	28
Park, (m/s)	4.76	5.15	5.57	5.88	6.32	6.84	7.40	8.00
KM, (m/s)	4.73	5.12	5.54	5.88	6.33	6.86	7.44	8.00

Table 12: Third row, $u_\infty = 8.0$ m/s

#M600	29	30	31	32	33	34	35
Park, (m/s)	5.15	5.57	5.89	6.32	6.84	7.40	8.00
KM, (m/s)	5.08	5.50	5.84	6.28	6.81	7.38	8.00

Table 13: Fourth row, $u_\infty = 8.0$ m/s

In the tables, it can be seen that the incident velocity for each M600 lie close to each other using both models, showing the validity of both approaches.
The total power generation related to the HAWT[8] and AWES systems using both models located in the OWEZ at free stream wind velocity $u_\infty = 8.0$ is denoted in the table below together with the corresponding power-to-mass ratio.

Model	Total power (MW)	power-to-mass (W/kg)
Vestas 90-3MW	22	1.54
Makani M600 (Park)	4.42	5.94
Makani M600 (KM)	4.43	5.94

Table 14: Power production OWEZ using Vestas 90-3MW or Makani M600, $u_\infty = 8.0$

The power-to-mass ratio here is especially important. With almost four times less resources used, the power output would be the same. With an eye on the future with limited resources, this could be a huge step towards a sustainable future, with renewable energy. These values were obtained for $u_\infty = 8.0$ as in this manner the values could be compared to valid values pictured in figure 9.

5.1.4 Scaling for a higher altitude

One of the reasons AWES could become feasible is because AWES could capture the wind speed at higher altitudes. Here the wind speeds are higher and therefore the power

[8] Estimation based on figure 9

generation could be higher as well[35][37]. A study by TNO recorded average wind speeds of 10.18 m/s at an altitude of 291 meters at the EPL platform in the North Sea[65]. Therefore the models are tested at a free stream wind velocity of $u_\infty = 10.18$. The incident velocities are displayed in the tables below.

#M600	0	1	2	3	4	5	6	7	8	9	10	11
Park, (m/s)	4.29	4.64	5.02	5.43	5.88	6.36	6.87	7.43	8.04	8.69	9.41	10.18
KM, (m/s)	4.32	4.66	5.04	5.45	5.89	6.37	6.89	7.46	8.08	8.75	9.49	10.18

Table 15: First row, $u_\infty = 10.18$ m/s

#M600	12	13	14	15	16	17	18	19	20
Park, (m/s)	5.57	6.02	6.52	7.05	7.43	8.04	8.70	9.41	10.18
KM, (m/s)	5.55	6.00	6.49	7.03	7.45	8.07	8.74	9.48	10.18

Table 16: Second row, $u_\infty = 10.18$ m/s

#M600	21	22	23	24	25	26	27	28
Park, (m/s)	6.06	6.56	7.09	7.49	8.04	8.70	9.41	10.18
KM, (m/s)	6.02	6.51	7.05	7.48	8.06	8.73	9.46	10.18

Table 17: Third row, $u_\infty = 10.18$ m/s

#M600	29	30	31	32	33	34	35
Park, (m/s)	6.56	7.09	7.49	8.04	8.70	9.41	10.18
KM, (m/s)	6.46	7.00	7.43	8.00	8.66	9.39	10.18

Table 18: Fourth row, $u_\infty = 10.18$ m/s

The total power generated by the wind farm, using both models, is then displayed in the table below, together with the power-to-mass ratio. The Vestas 90-3MW was not taken into account at this wind speed, as no other reliable data was found for the power generation.

Model	Total power (MW)	power-to-mass (W/kg)
Makani M600 (Park)	9.11	12.2
Makani M600 (KM)	9.12	12.3

Table 19: Power production OWEZ using Makani M600, $u_\infty = 10.18$ m/s

For an increase in free stream wind velocity of only 2.18 m/s, the power-to-mass ratio increases by a factor of more than 2. This shows the amount of uncaptured power generation that could be converted to electricity using AWES.

This simulation was done to simulate the wake effects generated by the Makani M600 at a wind direction angle which was least favourable for the power generation. In the next simulation, an optimised layout will be made for an offshore AWE farm.

5.2 Optimized layout

Whereas the Makani M600s were fully submerged in the wake of its downstream system, in the second simulation a more realistic wind farm will be reproduced. In this simulation, the M600s are placed such that the wake effects of the wind farm are minimized. Using a genetic algorithm (GA) the optimal locations of the AWES are found.

The genetic algorithm is a method to solve both constrained and unconstrained optimisation problems. It is a heuristic search algorithm designed for optimisation problems of two dimensions. Inherently genetic algorithms were designed for the process of natural selection and genetics of biological evolution [66]. The steps of the GA are listed below[67]:

1. **Initialisation :** First, a 'population' is created. The initial population consists of random layouts for wind farms generated randomly. This initial population is the first 'generation' of the GA.
2. **Evaluation:** Then the fitness of each wind farm is determined. They are evaluated through the total power output of the farm. The higher the total power, the higher the fitness score the farm is given.
3. **Selection:** The wind farms are selected based on their fitness score. This step determines which farms are eligible for reproduction.
4. **Crossover:** The individual wind farms are then combined using crossover techniques to generate new solutions or 'offspring'.
5. **Mutation:** To keep diversity, random mutations are introduced in the offspring. Change of the location of an individual system in the wind farm for example.
6. **Replacement:** The offspring then replaces some or all of the previous population, depending on the fitness score. These individual wind farms will move on to the next generation.
7. **Repeat:** Steps 2-6 are then repeated until the stopping criterion is met. In this case, after a hundred generations.

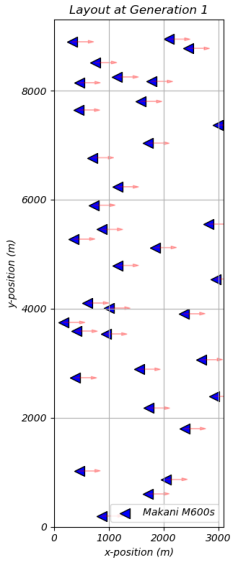
This algorithm is then applied to the Park model using the same dimensions as the OWEZ. The Kaufman-Martin is not used for this algorithm as this would require a larger computational capacity. The OWEZ has an area of 27 km^2[68], therefore the layout for a Makani M600 wind farm has comparable dimensions of 3000 $\times$ 9000 in meters. The wind direction is assumed to be from west to east. The population sizing doesn't have a general

rule. To determine this, a simple method is used to dimension the initial population. This method provided by Gatscha (2016)is given by[69]:

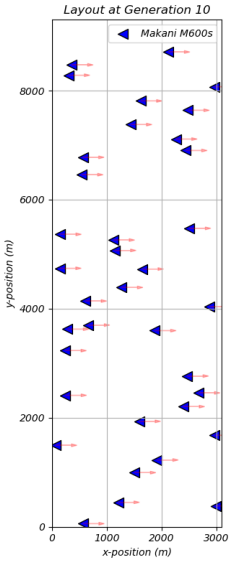
$$n = \frac{\#\text{Grids} * \#\text{Turbines}}{\text{Iteration}} \tag{71}$$

5.2.1 Free choice

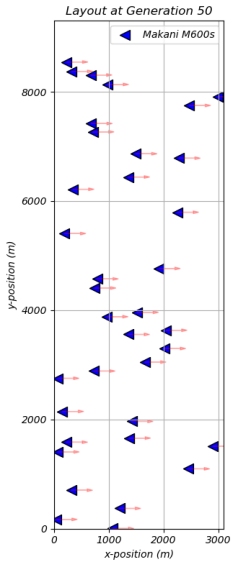
Whenever the M600s may be freely placed in the venture with dimensions 3000×9000 in meters, the grid is found by dividing the width by $2D$ and the height by D as the Park model is valid for downstream distances of $2D$[25] and the M600s must have a distance of at least D between them to prevent interference. The M600s can therefore be placed on $10 \times 60 = 600$ locations. The GA is computed using a hundred generations and 36 M600s. Therefore the population n equals 216. The layouts ensued by the GA are given at generation 1, 10, 50 and 100 below. Of every generation, the best layout is chosen and displayed.



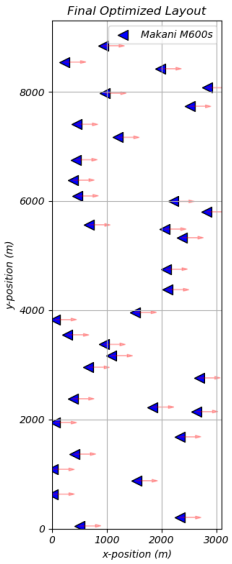
(a) Best layout after 1 generation; Total power = 19.14MW



(b) Best layout after 10 generations; Total power = 19.17MW



(c) Best layout after 50 generations; Total power = 19.25MW



(d) Best layout after 100 generations; Total power = 19.31MW

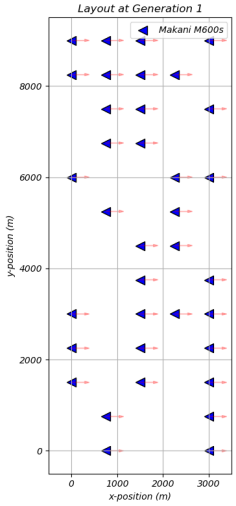
Figure 13: Optimal layouts using the GA after different generations; M600s freely placed

For free stream wind velocity of $u_\infty = 10.18$, the total power generation of the final optimized layout equals **19.31 MW**. Note that using the power equation 70, at a wind

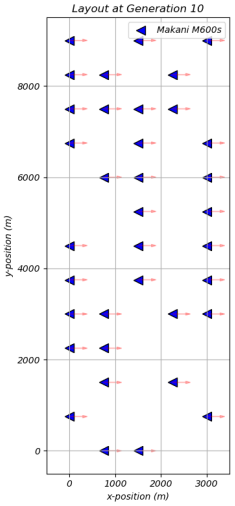
speed of $u_\infty = 10.18$, $P = 543$ KW for the individual M600. Multiply this by 36 results in a total power output of **19.54 MW**. This means that the total power output from the optimal layout is almost the same as its power capacity. This contradicts the claim that wake effects are not negligible in AWE. Then why is the output of the wind farm almost equal to the total power capacity? The answer to this can be reasoned in at least two ways.
Firstly, the wind is considered to only flow from west to east, as a result of which the GA places the M600s such that they are not in each other's wake by placing them on different y positions. When a wind distribution is introduced, the GA would find other optimal layouts, resulting in another total power output.
Secondly, the GA can now place the M600s at every coordinate in the area given. This allows placing the M600 at the positions where the wake effects are the least. Realistically, wind turbines in a conventional wind farm are placed on a grid as shown in figure 8.

5.2.2 Grid-constrained placement

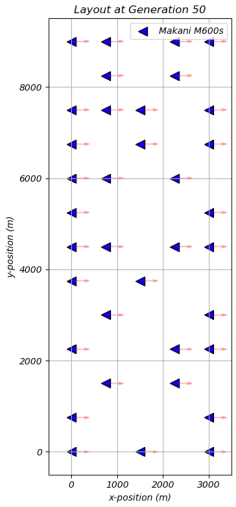
To depict a realistic wind farm, the optimal layout of the Makani M600 wind farm will be approached from a different point of view. In this layout, the M600s are constrained to a grid. To not interfere with the incident M600s the systems need to be at least $5D$ apart such that they don't affect each other with turbulence[70]. $5D$ equals about 750 meters for the Makani M600 as built. Therefore the grid cells are 750×750 meters in the x- and y-direction respectively. The system's location is constrained to the vertices of this grid. Again using the GA an altered optimal layout is found for the Makani M600 wind farm. Since the grid size is changed, so does the population by equation 71. #Grids now equals 65, therefore the population size now equals 24. The following layouts were found after 1, 10, 50 and 100 generations. Again, from every generation, the best layout is chosen and displayed:



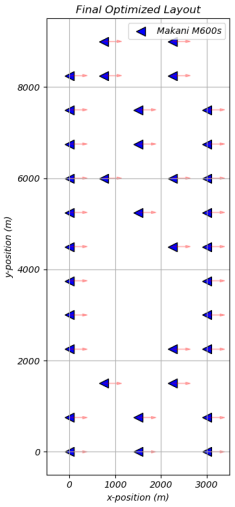
(a) Best layout after 1 generation; Total power = 16.50MW



(b) Best layout after 10 generations; Total power = 16.61MW



(c) Best layout after 50 generations; Total power = 16.81MW



(d) Best layout after 100 generations; Total power = 17.03MW

Figure 14: Optimal layouts using the GA after different generations; M600s placed on grid

Again for the free stream wind velocity of $u_\infty = 10.18$ the power generation for every layout was calculated. The total power output of the best layout after 100 generations equals **17.03 MW**. Compared to the method where the M600s were freely placed, this is a significant drop in power output.

5.2.3 Comparison & Outlook

For both methods, the power-to-mass ratio and total power generated is displayed in the below table, together with the power capacity of a Makani M600 wind farm.

	Total power (MW)	power-to-mass (W/kg)
Power capacity	19.54	26.2
GA (free)	19.32	26.0
GA (grid)	17.03	22.9

Table 20: Power production best Makani M600 farm layouts, $u_\infty = 10.18$ m/s

Comparing the GA for placement of the M600s in a wind farm to table 19 the power generated has more than or almost doubled together with the power-to-mass ratio. In table 19 the M600s were fully submerged in the wake generated by the system(s) downstream to themselves. The GA freely placed the M600s on a venture with the same area. The difference in power output portrays the influence between full-wake and partial-wake submergence for the AWES. It illustrates the significance of wake effects for airborne wind farms.

Furthermore the power-to-mass ratio in table 20 is interesting for future research of AWES application in wind farms. To put the results obtained from the GA in perspective, the comparison is made to a wind farm of 36 Vestas 90-3MW turbines. Since the mass of the Makani M600 is slightly more than 19 times lower than that of the Vestas 90-3MW. Suppose 19 more of these Makani M600 wind farms were built, they would have a total power output of between **324 MW** and **367 MW**, depending on whether the turbines are freely placed or placed on a grid respectively. This is more than 3 times the power capacity of the OWEZ, let alone the actual power output. A significant increase.

Figure 9 shows that the median power output of the OWEZ is roughly 32MW, which is interesting to compare with the layout created by the GA. For a free stream wind velocity of $u_\infty = 8.0$ the results are as follows:

	Total power (MW)	power-to-mass (W/kg)
Power capacity, M600	9.48	12.7
GA (free), M600	9.38	12.6
GA (grid), M600	8.22	11.0
OWEZ, Vestas	32	2.2

Table 21: Power production best Makani M600 farm layouts and median OWEZ, $u_\infty = 8.0$ m/s

For the same power output as the OWEZ, it would suffice to use four optimal Makani M600 wind farms using the grid-constrained GA. Using these four wind farms would take

up only 20% of the mass used in the OWEZ, therefore far less resource consumption for the same power output.

At first glance of table 20, the method where the GA is used to freely place the M600s looks preferable. This provides the highest power output and therefore should be the most viable solution. Despite the optimality of this solution, it is preferable to place the AWES on a grid. To lower costs, the amount of cable used for the power grid is to be minimized. Furthermore, for maintenance purposes, it is more efficient to build a patterned wind farm[71]. Consequently, both methods are suboptimal. Either the design is not efficient or the power generation is suboptimal. A middle ground must be found by combining both these methods.

6 Results

The goal of this research was to investigate the performance of AWES in wind farms against conventional HAWTs. First, the OWEZ was imitated using AWES, and the power output was then compared to the OWEZ using HAWTs. The results indicate a clear difference in performance between the AWES and the HAWTs. As expected the total power output of the Makani M600 wind farm was significantly lower compared to the Vestas 90-3MW wind farm under identical wind conditions. Using both the Park model (4.1) and the Kaufman-Martin model (4.2) for a free stream wind velocity of $u_\infty = 8.0$ with the turbines aligned with the wind direction (319°), the M600 array produced a total power output of order 4-6 times smaller than the HAWTs. This difference in power generation is explained by the fact that the power capacity of the Makani M600 (600 kW) and the Vestas 90-3MW (3 MW) differ by a factor of 5. Despite this, since the Vestas 90-3MW weighs roughly 19 times more than the M600, the M600 shows a remarkable advantage in terms of the power-to-mass ratio. When experiencing full wake effects the power-to-mass ratio of the M600 is a factor 3.9 times bigger than that of the Vestas turbine.

After the replication of the OWEZ using the M600 system, then an optimal layout was found using the GA with a fixed wind direction. Here the systems were not fully submerged in the wake of its downstream neighbour. The results of the GA were even more promising. The optimal, grid-constrained layout now had a power-to-mass ratio 5 times higher than that of OWEZ, using the median power production (32 MW) of the OWEZ. The change from 3.9 to 5 times better power-to-mass ratio is clarified by the lower shadowing factor γ_α of AWES compared to HAWTs. This highlights a key advantage for AWES. Far fewer resources are used for the same power output. This could translate into lower material costs and larger scalability opportunities for AWES.

The lower shadowing factor is a result of less wake interference by the AWES in a wind farm. The incident velocities of the freely placed M600 systems by the GA didn't differ much from the free stream wind velocity, as a result of which the power generation was nearly equal to the power capacity. This backs the earlier assumptions where the wake effects of AWES were treated as negligible. The results of the grid-constrained GA and OWEZ replication, however, indicate that the M600 systems underwent substantial wake effects. Especially in the OWEZ replication, both the Park and Kaufman-Martin models depicted strong velocity deficits caused by wake effects. In optimised layouts where the turbines could be placed on a grid or freely, the wake effects were reduced considerably. Here, for a free stream wind velocity of $u_\infty = 8.0$ the total power output was at least twice as high compared to the OWEZ replication. Overall, the comparison indicates that while the wake effects of AWES are not negligible, optimised layouts can mitigate the wake effects severely.

7 Conclusion

This research has found a way of optimizing the design layout of an offshore wind farm using AWES. After finding an analytical solution for the wake effects using the Park model and a numerical solution using the Kaufman-Martin model, the OWEZ, consisting of 36 Vestas 90-3MW turbines, was replicated using 36 Makani M600 systems. By implementation of these models, it was found that the wake effects generated by AWES are not negligible as they were previously thought to be. In comparison to the OWEZ, the performance of the Makani M600s in terms of the power-to-mass ratio was 3.9 times higher for a free stream wind velocity of $u_\infty = 8.0$.

The optimal layout was found using the Park model in a genetic algorithm. To be able to compare it to the OWEZ, the same dimensions of 3000×9000 meters were used. First, the algorithm was allowed to place the 36 M600s freely in the domain. Thereafter the M600s were only allowed to be placed on a grid since this is more common for real-life implementation of a wind farm. The power generated by the free algorithm and the grid-constrained algorithm are 19.32 MW and 17.03 MW respectively with a corresponding power-to-mass ratio of 26.0 W/kg and 22.9 W/kg. This was found for $u_\infty = 10.18$, which is the mean wind velocity at an altitude of 291 meters at the OWEZ site. For $u_\infty = 8.0$ the power generation is given by 9.38 MW and 8.22 MW respectively with corresponding power-to-mass ratios of 12.6 W/kg and 11.0 W/kg. These power-to-mass ratios were 5.7 and 5 times higher than that corresponding to the OWEZ for this wind speed. This means that for the same power output, only %20 of the resources are needed.

In conclusion, the wake effects of AWES are significant for a wind farm and therefore are not to be neglected. Also, since the power-to-mass ratio of the optimal M600 wind farm is five times higher than that of the OWEZ using Vestas 90-3MW turbines, this research has found that for future development it would be preferable to use AWES in a wind farm instead of the conventional HAWTs.

8 Discussion

8.1 Limitations

Although the AWES wind farm showed promising insights, there are several limitations that need to be discussed for validity of the research. Firstly, the parameters in table 1 for the Makani M600 as-built were developed for a minimal altitude of 110 meters. For results in 5.1 and 5.2 the average wind velocity of $u_\infty = 10.18$ at an altitude of about 300 meters. Therefore it could be possible that the parameters such as the drag and lift coefficients vary for different altitudes. However, since these are dependent on velocity instead of height, and the Makani M600 as-built has a rated wind speed of 11 m/s, the parameters shouldn't vary much.

Secondly, to simplify the configuration of the wakes generated by the AWES, it was assumed that the area of the flightpath and wake of a system was a perfect annulus. In reality the area of the flightpath of the airborne device is not a perfect annulus, however, this assumption came forth out of literature such as the Kaufman-Martin model [49][73]. Here the flightpath area and therefore the wake area is assumed to be a perfect annulus, validating the assumption made in this research.

In the entrainment-based Kaufman-Martin model the mass flow, mass flux in the wake region and mass flux in the core region are dependent on entrainment constant E. These variables are used to find results for the velocity deficit, inner wake diameter and outer diameter. To correctly find these values, a value must be assigned to E. However, this constant is based on empirical data, and does not have a general equation. Kaufman-Martin et al. (2022) states that typically $E = 0.15$ for HAWT models in low-turbulence inflow. This value however can reach up to 0.6 for high-turbulence flows. This then leaves E to be in a range of $0.15 \leq E \leq 0.6$. Since the Kaufman-Martin model exhibited roughly the same results as the Park model for $E = 0.31$, this value was chosen as the entrainment constant. While being grounded by the allowed range, the chosen value has not been empirically verified. Therefore the results of the Kaufman-Martin model must be viewed as indicative rather than explicit.

Lastly, the power equation 70, derived by Loyd (1980), is the upper bound of power generation for AWES. Therefore the results are likely to be overestimated for the actual power generation using the M600. Whereas in the modeling of this research optimal conditions were assumed, in reality these conditions vary a lot. These variations would reduce power generations. Additionally, perfect control and no failures were assumed for the airborne devices, ignoring potential losses. Since this research focused mainly on the wake effects of AWES and its performance in a wind farm, the situations where the conditions are not optimal only bring forth haziness. Using optimal conditions, the wake effects were captured the most clearly. Situations where these conditions vary were therefore out of the scope of this research.

8.2 Future research possibilities

Building on this study several possibilities for future work arise. To improve the optimized layout used in this research, and relate it to reality more, the wind distribution at the OWEZ site could be integrated. The GA now only considers a fixed wind direction from west to east, whereas in real situations the wind direction turns corresponding to its wind distribution. This could give a realistic indication of the annual energy production of a wind farm using AWES. Because of a lack of data, it was not implemented in this research.

In this research the Makani M600 model was used to model the wake effects of AWES. Exploring other models as airborne devices for an AWES wind farm would increase the understanding of wake effects for AWES. After implementation of different models, it could be determined which airborne device is the most suitable for airborne wind farms. Together with this only drag-mode AWES were considered. Exploration of lift-mode AWES in a wind farm could possibly present more feasible results than the drag-mode AWES. Exploring this could therefore be an interesting future research.

Lastly, this research did not incorporate the total cost or levelized cost of electricity, while this may be the most important variable for potential investors. While this study mostly discusses technical performance differences and possibilities for AWES compared to HAWTs, the actual development of AWES wind farms rely mostly on economic viability. This should include costs for wind development, maintenance, venture, monitoring and decommissioning, however these values are not yet available for AWES as it still is in an early phase of development. For HAWTs a general equation for the levelized cost of electricity, only for AWES, this is yet to come. Therefore future research considering the cost of an AWES wind farm would be very viable for the future of AWES and sustainable energy.

References

[1] https://www.technia.com/blog/which-renewable-energy-sources-are-most-reliable/#:~:text=At%20present%2C%20wind%20power%20is,without%20its%20challenges%20and%20disadvantages.

[2] https://www.sciencedirect.com/science/article/pii/S0960148111005155#:~:text=The%20two%20main%20eff

[3] https://home.uni-leipzig.de/energy/energy-fundamentals/15.htm

[4] Barthelmie, R. J., Pryor, S. C., Frandsen, S. T., Hansen, K. S., Schepers, J. G., Rados, K., Schlez, W., Neubert, A., Jensen, L. E., and Neckelmann, S.: Quantifying the Impact of Wind Turbine Wakes on Power Output at Offshore Wind Farms, J. Atmos. Ocean. Tech., 27, 1302–1317, https://doi.org/10.1175/2010JTECHA1398.1, 2010.

[5] https://wfo-global.org/wp-content/uploads/2024/04/WFO-Report-2024Q1.pdf

[6] https://www.statista.com/statistics/280704/world-power-consumption/

[7] https://www.sciencedirect.com/topics/engineering/airborne-wind-energy#: :text=Airborne%20wind%20energy%20(AWE)%20is,or%20more%20tethers%20%5B3%5D.

[8] Kruijff, M. and Ruiterkamp, R.: A Roadmap Towards Airborne Wind Energy in the Utility Sector, in: Airborne Wind Energy: Advances in Technology Development and Research, edited by: Schmehl, R., Springer Singapore, 643–662, https://doi.org/10.1007/978-981-10-1947-0, 2018.

[9] Echeverri, P., Fricke, T., Homsy, G., and Tucker, N.: The Energy Kite: Selected Results from the Design, Development, and Testing of Makani's Airborne Wind Turbines, Part I of III, Tech. rep., https://archive.org/details/theenergykite (last access: 19 May 2022), 2020.

[10] Malz, E., Zanon, M., and Gros, S.: A Quantification of the Performance Loss of Power Averaging in Airborne Wind Energy Farms, in: 2018 European Control Conference (ECC), https://doi.org/10.23919/ECC.2018.8550357, 2018.

[11] Roque, L. A., Paiva, L. T., Fernandes, M. C., Fontes, D. B., and Fontes, F. A.: Layout optimization of an airborne wind energy farm for maximum power generation, in: the 6th International Conference on Energy and Environment Research – Energy and environment: challenges towards circular economy, Energy Rep., 6, 165–171, https://doi.org/10.1016/j.egyr.2019.08.037, 2020.

[12] Leuthold, R., De Schutter, J., Malz, E., Licitra, G., Gros, S., and Diehl, M.: Operational Regions of a Multi-Kite AWE System, in: 2018 European Control Conference (ECC), 52–57, https://doi.org/10.23919/ECC.2018.8550199, 2018.

[13] Leuthold, R., Crawford, C., Gros, S., and Diehl, M.: Engineering Wake Induction Model For Axisymmetric Multi-Kite Systems, J. Phys.: Conf. Ser., 1256, 012009, https://doi.org/10.1088/1742-6596/1256/1/012009, 2019.

[14] De Lellis, M., Reginatto, R., Saraiva, R., and Trofino, A.: The Betz limit applied to Airborne Wind Energy, Renew. Energy, 127, 32–40, https://doi.org/10.1016/j.renene.2018.04.034, 2018.

[15] Gaunaa, M., Forsting, A. M., and Trevisi, F.: An engineering model for the induction of crosswind kite power systems, J. Phys.: Conf. Ser., 1618, 032010, https://doi.org/10.1088/1742-6596/1618/3/032010, 2020.

[16] Vermeer, L. J., Sørensen, J. N., & Crespo, A. (2003). Wind turbine wake aerodynamics. Progress in Aerospace Sciences, 39(6–7), 467–510. https://doi.org/10.1016/S0376-0421(03)00078-2

[17] Sanderse, B., van der Pijl, S. P., & Koren, B. (2011). Review of CFD for wind-turbine wake aerodynamics. Wind Energy, 14(7), 799–819. https://doi.org/10.1002/we.458

[18] https://www.grc.nasa.gov/www/k-12/airplane/nseqs.html

[19] https://www.ansys.com/simulation-topics/what-is-computational-fluid-dynamics#: :text=Computational%20fluid%20dynamics%20(CFD)%20is,mass%2C%20momentum%2C%20and'

[20] Niels Otto Jensen. A note on wind generator interaction. Number 2411 in Risø-M. 1983.

[21] I Katic, J Højstrup, and NO Jensen. A simple model for cluster efficiency. In European Wind Energy Association Conference and Exhibition, pages 407–410, 1986

[22] https://dictionary.cambridge.org/dictionary/english/entrainment#google_vignette

[23] Karakouzian, M. M., Kheiri, M., & Bourgault, F. (2022). A survey of two analytical wake models for crosswind kite power systems. Physics of Fluids, 34(9), 097111. https://doi.org/10.1063/5.0102388

[24] https://www.sciencedirect.com/science/article/pii/S026382232300661X?via%3Dihub

[25] John F Ainslie. Calculating the flowfield in the wake of wind turbines. Journal of Wind Engineering and Industrial Aerodynamics, 27(1):213–224, 1988.

[26] https://www.britannica.com/science/eddy-fluid-mechanics

[27] https://www.sciencedirect.com/topics/physics-and-astronomy/shear-layer/#: %3Atext=A%20shear%20layer%20is%20defined,in%20the%20ocean%20or%20atmosphere.

[28] https://www.sciencedirect.com/science/article/pii/S0960148119304239?via%3Dihub

[29] https://wes.copernicus.org/articles/7/1093/2022/wes-7-1093-2022.pdf

[30] https://www.copperpodip.com/post/airborne-wind-energy-awe-system-future-of-wind-energy

[31] https://wes.copernicus.org/preprints/wes-2024-139/

[32] https://www.copperpodip.com/post/airborne-wind-energy-awe-system-future-of-wind-energy

[33] Loyd, M. L. (1980). Crosswind kite power (for large-scale wind power production). Journal of Energy, 4(3), 106–111. https://doi.org/10.2514/3.48021

[34] Baayen, J. H., Ockels, W. J.: Tracking control with adaption of kites. IET Control Theory and Applications 6(2), 182–191 (2012). doi: 10.1049/iet-cta.2011.0037

[35] https://arxiv.org/pdf/1808.07718

[36] Diehl, M.: Airborne wind energy: Basic concepts and physical foundations, in: Airborne Wind Energy, edited by: Ahrens, U., Diehl, M., and Schmehl, R., Springer-Verlag, Berlin, Heidelberg, 3–22, https://doi.org/10.1007/978-3-642-39965-7_5, 2013.

[37] https://skysails-power.com/airborne-wind-fundamentals/#: %3Atext=Higher%20yields, %20most%20reliable%20and%20steady%20airflow.

[38] https://www.sciencedirect.com/science/article/pii/S2352484722001093#fig1

[39] https://www.sciencedirect.com/science/article/pii/S0960148118304427

[40] Michele Samorani. The wind farm layout optimization problem. In Handbook of Wind Power Systems, pages 21–38. Springer, 2013.

[41] Technical University of Denmark (DTU) Department of Wind Energy. Wasp 11 help facility and on-line documentation. chm file, 23 October 2014.

[42] https://www.sciencedirect.com/science/article/pii/S136403211600143X

[43] https://www.simscale.com/docs/simwiki/cfd-computational-fluid-dynamics/compressible-flow-vs-incompressible-flow/

[44] Wei Yu (2025), "Blade Element Momentum Theory", AE4135 Rotor/wake aerodynamics, Delft university of Technology.

[45] Vuik, C., Vermolen, F. J., van Gijzen, M. B., & Vuik, M. J. (2015). Numerical methods for ordinary differential equations (2nd ed.). TU Delft Open. https://doi.org/10.5074/t.2023.001

[46] https://pubs.aip.org/aip/pof/article/34/9/097111/2845803/A-survey-of-two-analytical-wake-models-for

[47] A Pena and O Rathmann. The atmospheric stability dependent infinite wind farm and wake decay coefficient. Wind Energy, in review, 2011.

[48] Jungchul Choi and Martin Shan. Advancement of Jensen (Park) wake model. In Proceedings of the European Wind Energy Conference and Exhibition, pages 1–8, 2013.

[49] Kaufman-Martin, S., Naclerio, N., May, P., and Luzzatto-Fegiz, P.: An entrainment-based model for annular wakes, with applications to airborne wind energy, Wind Energy, 25, 419–431, https://doi.org/10.1002/we.2679, 2022.

[50] Caulfield, C., & Luzzatto-Fegiz, P. (2018). An entrainment model for fully-developed wind farms: effects of atmospheric stability and an ideal limit for wind farm performance. https://doi.org/10.17863/CAM.33342

[51] Morton BR, Taylor GI, Turner JS. Turbulent gravitational convection from maintained and instantaneous sources. Proc R Soc Lond A Math Phys Sci. 1956; 234(1196): 1-23.

[52] https://www.cambridge.org/core/journals/journal-of-fluid-mechanics/article/entrainment-hypothesis-80-years-old-and-still-going-strong/01A28B98995042F5B0470F8492672EA3

[53] Ciri, U., & Salvetti, M. V. (2018). A one-parameter model for turbine wakes from the entrainment hypothesis. Journal of Physics: Conference Series, 1037(7), 072019. https://doi.org/10.1088/1742-6596/1037/7/072019

[54] Suraishkumar, G.K. (2014). Mass Flux. In: Continuum Analysis of Biological Systems. Biosystems & Biorobotics, vol 5. Springer, Berlin, Heidelberg. https://doi.org/10.1007/978-3-642-54468-2_2

[55] https://www.britannica.com/science/momentum

[56] https://math.libretexts.org/Bookshelves/Differential_Equations/Numerically_Solving_Ordinary_Differential_Ec

[57] Haas, T., & Meyers, J. (2017). Comparison study between wind turbine and power kite wakes. Journal of Physics: Conference Series, 854(1), 012019. https://doi.org/10.1088/1742-6596/854/1/012019

[58] https://www.energy-xprt.com/products/makani-model-m600-airborne-wind-energy-technology-623052

[59] https://en.wind-turbine-models.com/turbines/603-vestas-v90-3.0#: :text=The%20rated%20power%20of%20Vestas,V90%2D3.0%20is%2090%20m.

[60] Echeverri, P., Fricke, T., Homsy, G., & Tucker, N. (2020). The Energy Kite: Selected results from the design, development, and testing of Makani's airborne wind turbines (Part I). Makani Technologies LLC. https://x.company/projects/makani/#

[61] Echeverri, P., Fricke, T., Homsy, G., & Tucker, N. (2020). The Energy Kite: Selected results from the design, development, and testing of Makani's airborne wind turbines (Part II, Technical Artifacts). Makani Technologies LLC, https://x.company/projects/makani/#

[62] Larsen, T. J., Aagaard Madsen , H., Larsen, G. C., & Hansen, K. S. (2013). Validation of the dynamic wake meander model for loads and power production in the Egmond aan Zee wind farm. Wind Energy, 16(4), 605-624. https://doi.org/10.1002/we.1563

[63] https://www.renewables.ninja/

[64] https://ratedpower.com/glossary/rated-power/#: :text=Rated%20power%20definition%3A%20also%20known

[65] Vitulli, J. A., Eeckels, C. B. H., Bot, E. T. G., Verhoef, J. P., Bergman, G., & van der Werff, P. A. (2023). Offshore wind energy deployment in the North Sea by 2030: Long-term measurement campaign. EPL, 2016-2022 (TNO Report No. R10578). TNO. https://publications.tno.nl/publication/34640885/d83sPl/TNO-2023-R10578.pdf

[66] https://nl.mathworks.com/help/gads/what-is-the-genetic-algorithm.html

[67] https://www.datacamp.com/tutorial/genetic-algorithm-python

[68] https://www.noordzeeloket.nl/en/functions-use/offshore-wind-energy/free-passage-shared-use/offshore-wind-farm-egmond-aan-zee-owez/

[69] Gatscha, S. (2016). Generic optimization of a wind farm layout using a genetic algorithm (Master's thesis). University of Natural Resources and Life Sciences, Vienna. https://homepage.boku.ac.at/jschmidt/TOOLS/Masterarbeit_Gatscha.pdf

[70] https://renewablesfirst.co.uk/renewable-energy-technologies/windpower/community-windpower/location-size-no-of-wind-turbines/#: :text=The%20number%20of%20wind%20turbines,turbine%20i

[71] Tesauro, A., Réthoré, P.-E., & Larsen, G. C. (2012). State of the art of wind farm optimization. In Proceedings of EWEA 2012 - European Wind Energy Conference & Exhibition. European Wind Energy Association. https://backend.orbit.dtu.dk/ws/portalfiles/portal/7990594/State_of_the_Art_of_Wind_Farm_Optimization.pdf

[72] https://www.tudelft.nl/ewi/over-de-faculteit/afdelingen/electrical-sustainable-energy/dc-systems-energy-conversion-storage/research/wind-farm-design-and-optimization

[73] Mendonça, A. K. S., & Bornia, A. C. (2020). Electric power generation in wind farms with pumping kites: Levelized cost of energy and sensitivity analysis. Research, Society and Development, 9(7), e666974528. https://doi.org/10.33448/rsd-v9i7.4528

A Python code

A.1 OWEZ using Park model, analytically

```
import matplotlib.pyplot as plt
import numpy as np
import math
import pandas as pd
from geopy.distance import geodesic
from scipy.spatial.distance import cdist
#Parameters
rho = 1.225
u_inf = 10.18     #mean wind speed OWEZ
alpha = 0.038   #entrainment constant
D = 145         #outer diameter
d = 119.3       #inner diameter
Dr = 90         #Rotor diameter
z= 110          #altitude axisymmetric
z0 =0.0002      #surface roughness
C_T = 0.312     #coefficients
C_D = 0.312
C_L = 2.56
C_P = 0.45
A_wing = 32.9

OWEZ = pd.read_csv("/Users/user/Documents/BEP/OWEZ.csv")
#coordinates of the wind turbines scaled to be in [-90,90]
coordinates = list(zip(OWEZ['xpos']/100000,OWEZ['ypos']/100000))
met_mast = coordinates.pop()
# Create an empty distance matrix
num_turbines = len(coordinates)
distance_matrix = np.zeros((num_turbines, num_turbines))

# Compute pairwise distances
for i in range(num_turbines):
    for j in range(num_turbines):
        if i != j:  # Avoid zero distance for same point
            distance_matrix[i, j] = geodesic(coordinates[i], coordinates[j
    ]).meters  # Convert to meters

np.fill_diagonal(distance_matrix, np.inf)  # Ignore self-distances

# Convert lat/lon to a simple Cartesian (meters)
# Use one turbine as the origin
origin = coordinates[0]
def geo_to_cartesian(coord):
    # Approximation for small areas using geodesic distances
    x = geodesic((origin[1], origin[0]), (origin[1], coord[0])).meters  #
    East-West
    y = geodesic((origin[1], origin[0]), (coord[1], origin[0])).meters  #
    North-South
    if coord[0] < origin[0]:
        x = -x
    if coord[1] < origin[1]:
        y = -y
    return (x, y)

```

```
positions = np.array([geo_to_cartesian(c) for c in coordinates])

def rotate(positions):
    theta = np.arctan(positions[10][1]/positions[10][0])
    rotation_matrix = np.array([[np.cos(theta), np.sin(theta)],
                                [-np.sin(theta), np.cos(theta)]])
    return positions @ rotation_matrix.T

rotated_positions = rotate(positions) #now flow runs along x-axis


def integration_factor(x):
    term = d - 2* alpha *x
    if term!=0:
        return 0.5 * (1+term/abs(term))
    else:
        return 1


def shadow_factor(overlap, flightpath):
    return overlap/flightpath

def velocity_deficit(D, d, x, u_inc, overlap=1.0, flightpath=1.0):
    Dw = D + 2*alpha * x
    dw = integration_factor(x) * (d-2*alpha*x)
    term = 1- C_T
    numerator = 2*shadow_factor(overlap,flightpath)*(1-np.sqrt(term))*(D**2
     - d**2)
    denominator = Dw**2 - dw**2
    return numerator/denominator

reversed_positions = np.array(rotated_positions[::-1]) #reversed so that
    upstream comes first

def velocity_at_turbines(reversed_positions):
    N = len(reversed_positions)
    u_inc = np.full(N,u_inf) #initialize with freestream velocity
    #calculate per row, no wake effects between rows
    for i in range(1,7):
        dx = abs(reversed_positions[i][0]-reversed_positions[i-1][0])
        du = velocity_deficit(D, d, dx, u_inc[i])
        u_inc[i]=(1-du)*u_inc[i-1]

    for j in range(8,15):
        dx = abs(reversed_positions[j][0]-reversed_positions[j-1][0])
        du = velocity_deficit(D, d, dx, u_inc[j])
        u_inc[j]=(1-du)*u_inc[j-1]

    for k in range(16,24):
        dx = abs(reversed_positions[k][0]-reversed_positions[k-1][0])
        du = velocity_deficit(D, d, dx, u_inc[k])
        u_inc[k]=(1-du)*u_inc[k-1]

    for l in range(25,36):
        dx = abs(reversed_positions[l][0]-reversed_positions[l-1][0])
        du = velocity_deficit(D, d, dx, u_inc[l])
        u_inc[l]=(1-du)*u_inc[l-1]

    reverse_u_inc = np.array(u_inc[::-1])#reversed again such that turbines
```

```
     are in order
    return reverse_u_inc
incident_velocities = velocity_at_turbines(reversed_positions)
print(incident_velocities)


def power_equation(u_inc):
    powers = []

    for u in u_inc:
        P=(2/27)*rho*A_wing*C_L *(C_L/C_D)**2*u**3
        powers.append(P)

    return np.array(powers)

def total_power(u_inc):
    total = 0
    power_per_system=power_equation(u_inc)
    for p in power_per_system:
        total +=p
    return total
x=[u_inf]
print('Power AWES:',total_power(incident_velocities))

lst= list(range(len(reversed_positions)))
plt.bar(lst, velocity_at_turbines(reversed_positions), label = fr"$u_\infty
    ={u_inf}$")
plt.xlabel("#Turbines")
plt.ylabel('Incident velocity in m/s')
plt.ylim(3.0, 9)
plt.title('Wake effect')
plt.legend()
plt.grid(True)
plt.show()
#%%
closest_turbines = np.argmin(distance_matrix, axis=1)
# Print closest turbine for each turbine
latitudes = OWEZ['ypos'].to_numpy()/100000
longitudes = OWEZ['xpos'].to_numpy()/100000
latitudes = np.delete(latitudes,36)
longitudes = np.delete(longitudes,36)
# Create a scatter plot
plt.figure(figsize=(8, 6))
plt.scatter(longitudes, latitudes, c='b', marker='o', label='Turbines')
# Label each turbine
for i, (lon, lat) in enumerate(zip(longitudes, latitudes)):
   plt.text(lon, lat, str(i), fontsize=9, ha='right', color='red')
plt.xlabel("Longitude")
plt.ylabel("Latitude")
plt.title("Wind Turbine Layout (Geographic Coordinates)")
plt.legend()
plt.grid(True)
plt.show()
```

A.2 OWEZ using Kaufman-Martin, numerically

```
1
2  import numpy as np
3  import matplotlib.pyplot as plt
4  import pandas as pd
5  from geopy.distance import geodesic
6  from scipy.spatial.distance import cdist
7
8  rho = 1.225
9  t0 = 0
10 D = 145          #outer diameter
11 d = 119.3
12 E = 0.31
13 Vinf = 10.18
14 z= 110          #altitude axisymmetric
15 z0 =0.0002      #surface roughness
16 C_T = 0.312      #coefficients
17 C_D = 0.312
18 C_L = 2.56
19 A_wing = 32.9
20 Dw0 = np.sqrt(2*D**2-d**2)
21 dw0 = d
22 Vw0 = Vinf /3
23 mw0 = (1/4)*(Dw0 **2 - dw0**2)*Vw0
24 Mw0 = (1/4)*(Dw0 **2 - dw0**2)*Vw0**2
25 mi0 = dw0**2 * Vinf/4
26 num_steps = 10000
27 h = 0.1   #h*num_steps = 1000!
28
29
30 # Function representing the differential equation: dP/dt = k * P
31 def entrainment1(t,mw,Mw,mi,E,Vinf):
32     return 2*E*(Vinf-Mw/mw)*(np.sqrt(mi/Vinf+mw**2/Mw) +np.sqrt(mi/Vinf))
33 def entrainment2(t,mw,Mw,mi,E,Vinf):
34     return Vinf*(2*E*(Vinf-Mw/mw)*(np.sqrt(mi/Vinf+mw**2/Mw) +np.sqrt(mi/
       Vinf)))
35 def entrainment3(t,mw,Mw,mi,E,Vinf):
36     return -2*E*(Vinf-Mw/mw)*np.sqrt(mi/Vinf)
37
38
39
40
41
42 # Fourth-order Runge-Kutta method
43 def runge_kutta_4(f1,f2,f3, t0, mw0, h, num_steps, Mw0,mi0,E,Vinf):
44     solution = [(t0, mw0,Mw0,mi0)]
45     solution_scaled = [(t0,0,0,0)]
46     t = t0
47     mw = mw0
48     Mw = Mw0
49     mi = mi0
50
51     for _ in range(num_steps):
52         k1 = h * f1(t, mw,Mw,mi,E,Vinf)
53         k2 = h * f1(t + 0.5 * h, mw + 0.5 * k1,Mw,mi,E,Vinf)
54         k3 = h * f1(t + 0.5 * h, mw + 0.5 * k2,Mw,mi,E,Vinf)
55         k4 = h * f1(t + h, mw + k3,Mw,mi,E,Vinf)
56         g1 = h * f2(t, mw,Mw,mi,E,Vinf)
57         g2 = h * f2(t + 0.5 * h, mw,Mw + 0.5 * g1,mi,E,Vinf)
58         g3 = h * f2(t + 0.5 * h, mw,Mw + 0.5 * g2,mi,E,Vinf)
```

```
        g4 = h * f2(t + h, mw,Mw+ g3,mi,E,Vinf)
        l1 = h * f3(t, mw,Mw,mi,E,Vinf)
        l2 = h * f3(t + 0.5 * h, mi ,Mw,mi+ 0.5 * l1,E,Vinf)
        l3 = h * f3(t + 0.5 * h, mi ,Mw,mi+ 0.5 * l2,E,Vinf)
        l4 = h * f3(t + h, mi ,Mw,mi+ l3,E,Vinf)


        mw = mw + (k1 + 2 * k2 + 2 * k3 + k4) / 6
        Mw = Mw + (g1 + 2 * g2 + 2 * g3 + g4) / 6
        mi = mi + (l1 + 2 * l2 + 2 * l3 + l4) / 6
        t = t + h

        #print(mw)
        #print(Mw)
        #print(mi)

        #Terug schalen
        Vw = Mw/mw
        dw = 2* np.sqrt(mi/Vinf)
        Dw = 2* np.sqrt((mi/Vinf)+(mw**2/Mw))
        solution.append((t, mw,Mw,mi)) #om het in een uiteindelijke vector / matrix op te slaan.
        solution_scaled.append((t,Vw,dw,Dw)) #om het in een uiteindelijke vector / matrix op te slaan.

    return solution_scaled




# Solve the differential equation using Runge-Kutta method
solution_scaled = runge_kutta_4(entrainment1,entrainment2,entrainment3, t0, mw0, h, num_steps, Mw0,mi0,E,Vinf)

# Extracting time and population values for plotting
t_values = [t for t, _, _, _ in solution_scaled]
Vw_values = [Vw for _, Vw, _, _ in solution_scaled]
dw_values = [dw for _, _, dw, _ in solution_scaled]
Dw_values = [Dw for _, _, _, Dw in solution_scaled]

OWEZ = pd.read_csv("/Users/user/Documents/BEP/OWEZ.csv")
#coordinates of the wind turbines scaled to be in [-90,90]
coordinates = list(zip(OWEZ['xpos']/100000,OWEZ['ypos']/100000))
met_mast = coordinates.pop()
# Create an empty distance matrix
num_turbines = len(coordinates)
distance_matrix = np.zeros((num_turbines, num_turbines))

# Compute pairwise distances
for i in range(num_turbines):
    for j in range(num_turbines):
        if i != j:  # Avoid zero distance for same point
            distance_matrix[i, j] = geodesic(coordinates[i], coordinates[j]).meters  # Convert to meters

np.fill_diagonal(distance_matrix, np.inf)  # Ignore self-distances

# Convert lat/lon to a simple Cartesian (meters)
# Use one turbine as the origin
```

```
origin = coordinates[0]
def geo_to_cartesian(coord):
    # Approximation for small areas using geodesic distances
    x = geodesic((origin[1], origin[0]), (origin[1], coord[0])).meters  # East-West
    y = geodesic((origin[1], origin[0]), (coord[1], origin[0])).meters  # North-South
    if coord[0] < origin[0]:
        x = -x
    if coord[1] < origin[1]:
        y = -y
    return (x, y)

positions = np.array([geo_to_cartesian(c) for c in coordinates])

def rotate(positions):
    theta = np.arctan(positions[10][1]/positions[10][0])
    rotation_matrix = np.array([[np.cos(theta), np.sin(theta)],
                                [-np.sin(theta), np.cos(theta)]])
    return positions @ rotation_matrix.T

rotated_positions = rotate(positions) #now flow runs along x-axis
reversed_positions = np.array(rotated_positions[::-1])

def numerical_velocity_deficit(reversed_positions, t0, mw0, h, num_steps, Mw0, mi0, E, Vinf,dw0, Dw0):
    N = len(reversed_positions)
    u_list = np.full(N,Vinf)
    mw = mw0
    Mw = Mw0
    mi = mi0#initialize with freestream velocity
    #calculate per row, no wake effects between rows
    Vw = Vinf
    for i in range(1,7):
        dx = abs(reversed_positions[i][0]-reversed_positions[i-1][0])
        steps= int(dx/h -1)
        solution_scaled = runge_kutta_4(entrainment1, entrainment2, entrainment3, t0, mw, h, num_steps, Mw, mi, E, Vw)
        Vw = solution_scaled[steps][1]
        u_list[i] =Vw
        step_indices=list(range(num_steps+1))
        Vw_step = [solution_scaled[a][1]for a in step_indices]

        plt.plot([s/10 for s in step_indices], Vw_step, label = f"Makani M600 {35-i}")
        plt.plot(step_indices[steps]/10, Vw_step[steps], 'kv')

        mw = (1/4)*(Dw0 **2 - dw0 **2)*(Vw/3)
        Mw = (1/4)*(Dw0 **2 - dw0 **2)*(Vw/3)**2


    plt.xlabel('x in meters')
    plt.ylabel('Incident velocity in m/s')
    plt.title(r'Wake effect for $u_\infty = 8.0$')
    plt.legend()
    plt.grid(True)
    plt.show()
    Vw = Vinf
    for j in range(8,15):
```

```
        dx = abs(reversed_positions[j][0]-reversed_positions[j-1][0])
        steps = int(dx/h -1)
        solution_scaled = runge_kutta_4(entrainment1, entrainment2,
    entrainment3, t0, mw, h, num_steps, Mw, mi, E, Vw)
        Vw = solution_scaled[steps][1]
        u_list[j] =Vw

        step_indices=list(range(num_steps+1))
        Vw_step = [solution_scaled[a][1]for a in step_indices]

        plt.plot([s/10 for s in step_indices], Vw_step, label = f"Makani
    M600 {35-j}")
        plt.plot(step_indices[steps]/10, Vw_step[steps], 'kv')

        mw = (1/4)*(Dw0 **2 - dw0 **2)*(Vw/3)
        Mw = (1/4)*(Dw0 **2 - dw0 **2)*(Vw/3)**2

    plt.xlabel('x in meters')
    plt.ylabel('Incident velocity in m/s')
    plt.title(r'Wake effect for $u_\infty = 8.0$')
    plt.legend()
    plt.grid(True)
    plt.show()
    Vw = Vinf
    for k in range(16,24):
        dx = abs(reversed_positions[k][0]-reversed_positions[k-1][0])
        steps = int(dx/h -1)
        solution_scaled = runge_kutta_4(entrainment1, entrainment2,
    entrainment3, t0, mw, h, num_steps, Mw, mi, E, Vw)
        Vw = solution_scaled[steps][1]
        u_list[k] =Vw

        step_indices=list(range(num_steps+1))
        Vw_step = [solution_scaled[a][1]for a in step_indices]

        plt.plot([s/10 for s in step_indices], Vw_step, label = f"Makani
    M600 {35-k}")
        plt.plot(step_indices[steps]/10, Vw_step[steps], 'kv')

        mw = (1/4)*(Dw0 **2 - dw0 **2)*(Vw/3)
        Mw = (1/4)*(Dw0 **2 - dw0 **2)*(Vw/3)**2

    plt.xlabel('x in meters')
    plt.ylabel('Incident velocity in m/s')
    plt.title(r'Wake effect for $u_\infty = 8.0$')
    plt.legend()
    plt.grid(True)
    plt.show()
    Vw = Vinf
    for l in range(25,36):
        dx = abs(reversed_positions[l][0]-reversed_positions[l-1][0])
        steps = int(dx/h -1)
        solution_scaled = runge_kutta_4(entrainment1, entrainment2,
    entrainment3, t0, mw, h, num_steps, Mw, mi, E, Vw)
        Vw = solution_scaled[steps][1]
        u_list[l] =Vw

        step_indices=list(range(num_steps+1))
        Vw_step = [solution_scaled[a][1]for a in step_indices]
```

```
222
223         plt.plot([s/10 for s in step_indices], Vw_step, label = f"Makani
        M600 {35-l}")
224         plt.plot(step_indices[steps]/10, Vw_step[steps], 'kv')
225
226         mw = (1/4)*(Dw0 **2 - dw0 **2)*(Vw/3)
227         Mw = (1/4)*(Dw0 **2 - dw0 **2)*(Vw/3)**2
228
229     plt.xlabel('x in meters')
230     plt.ylabel('Incident elocity in m/s')
231     plt.title(r'Wake effect for $u_\infty = 8.0$')
232     plt.legend()
233     plt.grid(True)
234     plt.show()
235     return np.array(u_list[::-1])
236 incident_velocities = numerical_velocity_deficit(reversed_positions, t0,
        mw0, h, num_steps, Mw0, mi0, E, Vinf, dw0, Dw0)
237 def power_equation(u_inc):
238     powers = []
239
240     for u in u_inc:
241         P=(2/27)*rho*A_wing*C_L *(C_L/C_D)**2*(u**3)
242         powers.append(P)
243
244     return np.array(powers)
245 def total_power(u_inc):
246     total = 0
247     power_per_system=power_equation(u_inc)
248     for p in power_per_system:
249         total +=p
250     return total
251 print(incident_velocities)
252 print('Power AWES RK4:', total_power(incident_velocities))
```

A.3 Optimized layout, Park model, no constraints

```
1  import matplotlib.pyplot as plt
2  import numpy as np
3  import math
4  import pandas as pd
5  from geopy.distance import geodesic
6  from scipy.spatial.distance import cdist
7  import random
8  #Parameters
9  rho = 1.225
10 u_inf = 10.18     #mean wind speed OWEZ
11 alpha = 0.038    #entrainment constant
12 D = 145           #outer diameter
13 d = 119.3         #inner diameter
14 z= 100           #altitude axisymmetric
15 z0 =0.0002       #surface roughness
16 C_T = 0.312       #coefficients
17 C_D = 0.312
18 C_L = 2.56
19 A_wing = 32.9
20
```

```
21
22 def integration_factor(x):
23     term = d - 2* alpha *x
24     if term!=0:
25         return 0.5 * (1+term/abs(term))
26     else:
27         return 1
28
29
30
31 def gamma(i_pos, j_pos, x):
32     # Circle radii
33     R = D / 2  # outer radius flight path
34     r = d /2   # inner radius flight path
35     Rw = (D + 2 * alpha * x) / 2  # wake diameter at x
36     rw = max(0,integration_factor(x)*(d-2*alpha*x)/2) #inner wake diameter
       at x
37     # Center distance (in 2D)
38     dist = np.linalg.norm(np.array(j_pos) - np.array([j_pos[0], i_pos[1]]))
         # Only y-distance matters
39     def intersection(Ra, Rb):
40         if dist >= Ra +Rb:
41             return 0.0
42         if dist <= abs(Ra - Rb):
43             return 1.0
44         term1 = Ra**2 * np.arccos((dist**2 + Ra**2 - Rb**2) / (2 * dist *
       Ra))
45         term2 = Rb**2 * np.arccos((dist**2 + Rb**2 - Ra**2) / (2 * dist *
       Rb))
46         term3 = 0.5 * np.sqrt((-dist + Ra + Rb) * (dist + Ra - Rb) * (dist
       - Ra + Rb) * (dist + Ra + Rb))
47         return term1 + term2 - term3
48     if intersection(R,Rw) == 1.0:
49         return 1.0
50     elif intersection(R,Rw) == 0.0:
51         return 0.0
52     else:
53         A_overlap = (intersection(R,Rw) - intersection(R,rw) - intersection
       (r, Rw)+intersection(r,rw))
54         A_flightpath = np.pi*(R**2-r**2)
55         return A_overlap / A_flightpath
56
57 def velocity_deficit(D, d, x, u_inc, gamma, i_pos, j_pos):
58     Dw = D + 2*alpha * x
59     dw = integration_factor(x) * (d-2*alpha*x)
60     term = 1- C_T
61
62     numerator = 2*gamma(i_pos,j_pos, x)*(1-np.sqrt(term))*(D**2 - d**2)
63     denominator = Dw**2 - dw**2
64     return numerator/denominator
65
66 def in_wake(i_pos, j_pos):
67     return j_pos[0]>i_pos[0]
68
69 def delta_x(i_pos,j_pos):
70     return abs(j_pos[0]-i_pos[0])
71
72 def incident_velocities(layout):
73     N = len(layout)
```

```
    u_inc=np.full(N, u_inf)

    for j in range(N):
        deficit = []

        for i in range(N):
            dx = delta_x(layout[i],layout[j])
            if i == j:
                continue
            if in_wake(layout[i], layout[j]):
                overlap = gamma(layout[i], layout[j], dx)
                du = velocity_deficit(D, d, dx, u_inc[i], gamma, layout[i], layout[j])
                deficit.append(du)

        if deficit:
            total_deficit = np.sqrt(np.sum(np.square(deficit)))
            u_inc[j] = u_inf * (1-total_deficit)
    return u_inc

def power_equation(u_inc):
    powers = []

    for u in u_inc:
        P=(2/27)*rho*A_wing*C_L *(C_L/C_D)**2*u**3
        powers.append(P)

    return np.array(powers)
def total_power(u_inc):
    total = 0
    power_per_system=power_equation(u_inc)
    for p in power_per_system:
        total +=p
    return total

def random_layout(n, width, height, D):
    layout =[]
    attempts = 0
    max_attempt = 10000
    while len(layout) < n and attempts<max_attempt:
        x=np.random.uniform(0,width)
        y=np.random.uniform(0,height)
        position = np.array([x,y])

        if all(abs(y-s[1])>= D or abs(x-s[0])>=2*D for s in layout) :
            layout.append(position)
        attempts+=1
    if len(layout)<n:
        print('Not al turbines could be located')

    return np.array(layout)

#%%
def fitness(layout):
    u_inc = incident_velocities(layout)
    return total_power(u_inc)

def initial_pop(popsize, n ,width , height, D):
    return [random_layout(n, width, height, D) for _ in range(popsize)]
```

```
132
133 def constraint(layout, width, height, D):
134     sorted_layout = sorted(layout, key = lambda x: x[1])
135     filtered = [sorted_layout[0]]
136     for point in sorted_layout[1:]:
137         if all(abs(point[1] - p[1]) >= D or abs(point[0]-p[0])>=2*D for p
    in filtered):
138             filtered.append(point)
139
140     if len(filtered)<len(layout):
141         return random_layout(n,width, height, D)
142
143     return np.array(filtered)
144
145 def crossover(farm1, farm2):
146     n = len(farm1)
147     offspring=[]
148     for i in range(n):
149         if np.random.rand()<0.5: #uniform probability
150             offspring.append(farm1[i])
151         else:
152             offspring.append(farm2[i])
153     return constraint(np.array(offspring), width, height, D)
154
155 def mutate(layout, rate, width, height):
156     copy = layout.copy()
157
158     for i in range(len(copy)):
159         if np.random.rand()<rate:
160             xnew = np.random.uniform(0, width)
161             ynew = np.random.uniform(0, height)
162             copy[i]=[xnew,ynew]
163     return constraint(copy, width, height, D)
164
165 def genetic_algorithm(gen, popsize, rate, n, width, height, D):
166     pop= initial_pop(popsize, n, width, height, D)
167     best_fit = 0
168     best_layout = None
169     saved_layout ={}
170     for g in range(gen): #fitness
171         scores = [fitness(layout) for layout in pop]
172         best_index= np.argmax(scores)
173         if scores[best_index] > best_fit:
174             best_fit = scores[best_index]
175             best_layout = pop[best_index]
176             print(f"Gen {g+1}: Best Power = {scores[best_index]/1e6:.2f} MW
    ")
177         if g in [0,9,49]:
178             saved_layout[g] = pop[best_index]
179         #selection = top 25%
180         sorted_pop = [x for _, x in sorted(zip(scores,pop), key = lambda
    pair: pair[0], reverse = True)]
181         top = sorted_pop[:popsize // 4]
182
183         #make new population
184         new_pop = top.copy()
185         while len(new_pop) < popsize:
186             index = np.random.choice(len(top),2) #choose random parents
    from top
```

```
            f1 = top[index[0]]
            f2 = top[index[1]]
            offspring = crossover(f1,f2)
            offspring = mutate(offspring, rate, width, height)
            new_pop.append(offspring)

        pop = new_pop

    return best_layout, best_fit, saved_layout

gen=100
popsize =216
rate =0.1
n = 36
width = 3000
height = 9000

bestlayout, bestfitness, saved_layout = genetic_algorithm(gen, popsize,
    rate, n, width, height, D)
print(f"Final Best Power Output: {bestfitness/1e6:.2f} MW")

inc = incident_velocities(bestlayout)
print(inc)

x, y = bestlayout[:, 0], bestlayout[:, 1]

def plot_layout(layout, title, width, height):
    x, y = layout[:, 0], layout[:, 1]
    plt.figure(figsize=(8, 8))
    plt.scatter(x, y, c='blue', s=100, edgecolors='black', marker='<',
    label='Makani M600s')

    for i in range(len(layout)):
        plt.arrow(x[i], y[i], 2*D, 0, head_width=50, head_length=100, fc='
    red', ec='red', alpha=0.3)

    plt.xlim(0, width + 100)
    plt.ylim(0, height + 300)
    plt.gca().set_aspect('equal')
    plt.grid(True)
    plt.xlabel("x-position (m)")
    plt.ylabel("y-position (m)")
    plt.title(title)
    plt.legend()
    plt.tight_layout()
    plt.show()

plot_layout(saved_layout[0], "Layout at Generation 1", width, height)
plot_layout(saved_layout[9], "Layout at Generation 10", width, height)
plot_layout(saved_layout[49], "Layout at Generation 50", width, height)
plot_layout(bestlayout, "Final Optimized Layout", width, height)
```

A.4 Optimized layout, Park model, grid-constrained

```
import matplotlib.pyplot as plt
import numpy as np
```

```
import math
import pandas as pd
from geopy.distance import geodesic
from scipy.spatial.distance import cdist
import random
#Parameters
rho = 1.225
u_inf = 8.0    #mean wind speed OWEZ
alpha = 0.038   #entrainment constant
D = 145          #outer diameter
d = 119.3        #inner diameter
z= 100          #altitude axisymmetric
z0 =0.0002      #surface roughness
C_T = 0.312      #coefficients
C_D = 0.312
C_L = 2.56
A_wing = 32.9

gen=100
popsize =24
rate =0.1
n = 36
width = 3000
height = 9000



def integration_factor(x):
    term = d - 2* alpha *x
    if term!=0:
        return 0.5 * (1+term/abs(term))
    else:
        return 1



def gamma(i_pos, j_pos, x):
    # Circle radii
    R = D / 2  # outer radius flight path
    r = d /2   # inner radius flight path
    Rw = (D + 2 * alpha * x) / 2  # wake diameter at x
    rw = max(0,integration_factor(x)*(d-2*alpha*x)/2) #inner wake diameter
    at x
    # Center distance (in 2D)
    dist = np.linalg.norm(np.array(j_pos) - np.array([j_pos[0], i_pos[1]]))
      # Only y-distance matters

    def intersection(Ra, Rb):
        if dist >= Ra +Rb:
            return 0.0
        if dist <= abs(Ra - Rb):
            return 1.0
        term1 = Ra**2 * np.arccos((dist**2 + Ra**2 - Rb**2) / (2 * dist *
    Ra))
        term2 = Rb**2 * np.arccos((dist**2 + Rb**2 - Ra**2) / (2 * dist *
    Rb))
        term3 = 0.5 * np.sqrt((-dist + Ra + Rb) * (dist + Ra - Rb) * (dist
    - Ra + Rb) * (dist + Ra + Rb))
        return term1 + term2 - term3
```

```
    if intersection(R,Rw) == 1.0:
        return 1.0
    elif intersection(R,Rw) == 0.0:
        return 0.0
    else:
        A_overlap = (intersection(R,Rw) - intersection(R,rw) - intersection(r, Rw)+intersection(r,rw))
        A_flightpath = np.pi*(R**2-r**2)
        return A_overlap / A_flightpath if A_flightpath >0 else 0.0

def velocity_deficit(D, d, x, u_inc, gamma, i_pos, j_pos):
    Dw = D + 2*alpha * x
    dw = integration_factor(x) * (d-2*alpha*x)
    term = 1- C_T

    numerator = 2*gamma(i_pos,j_pos, x)*(1-np.sqrt(term))*(D**2 - d**2)
    denominator = Dw**2 - dw**2
    return numerator/denominator

def delta_x(i_pos,j_pos):
    return abs(j_pos[0]-i_pos[0])

def incident_velocities(layout):
    N = len(layout)
    u_inc=np.full(N, u_inf)
    for j in range(N):
        deficit = []

        for i in range(N):
            x = delta_x(layout[i],layout[j])
            if i == j:
                continue
            else:
                du = velocity_deficit(D, d, x, u_inc[i], gamma, layout[i], layout[j])
                deficit.append(du)

        if deficit:
            total_deficit = np.sqrt(np.sum(np.square(deficit)))
            u_inc[j] = u_inf * (1-total_deficit)

    return u_inc

def power_equation(u_inc):
    powers = []

    for u in u_inc:
        P=(2/27)*rho*A_wing*C_L *(C_L/C_D)**2*u**3
        powers.append(P)

    return np.array(powers)
def total_power(u_inc):
    total = 0
    power_per_system=power_equation(u_inc)
    for p in power_per_system:
        total +=p
    return total


```

```
def grid_point_generator(width, height, xcell, ycell):
    x_points = np.arange(0,width+1, xcell)
    y_points = np.arange(0,height+1,ycell)
    grid_points = np.array([[x,y] for x in x_points for y in y_points])
    return grid_points

def random_grid_layout(n, grid_points, D):
    layout =[]
    attempts = 0
    max_attempt = 10000
    while len(layout) < n and attempts<max_attempt:
        point = grid_points[np.random.randint(len(grid_points))]

        if all(abs(point[0] - p[0]) >=2*D or abs(point[1]-p[1])>=D for p in
 layout):
            layout.append(point)
        attempts+=1
    if len(layout)<n:
        print('Not al turbines could be located')

    return np.array(layout)


#%%
def fitness(layout):
    u_inc = incident_velocities(layout)
    return total_power(u_inc)

def initial_pop(popsize, n , grid_points, D):
    return [random_grid_layout(n, grid_points, D) for _ in range(popsize)]

def back_to_grid(point, grid_points): #back to nearest grid point
    dist = np.linalg.norm(grid_points - point, axis=1)
    return grid_points[np.argmin(dist)]

def constraint(layout, grid_points, D):
    filtered = []
    for point in layout:
        nearest_point = back_to_grid(point, grid_points)
        if all(abs(nearest_point[0]-p[0])>= 2*D or abs(nearest_point[1]-p
    [1])>=D for p in filtered):
            filtered.append(nearest_point)
    if len(filtered)<len(layout):
        return random_grid_layout(len(layout), grid_points, D)
    return np.array(filtered)

def crossover(farm1, farm2, grid_points):
    n = len(farm1)
    offspring=[]
    for i in range(n):
        if np.random.rand()<0.5: #uniform probability
            offspring.append(farm1[i])
        else:
            offspring.append(farm2[i])
    return constraint(np.array(offspring), grid_points, D)

def mutate(layout, rate, grid_points):
    copy = layout.copy()

```

```
    for i in range(len(copy)):
        if np.random.rand()<rate:
            copy[i] = grid_points[np.random.randint(len(grid_points))]
    return constraint(copy, grid_points, D)

grid_points = grid_point_generator(width, height, 750, 750)
def genetic_algorithm(gen, popsize, rate, n, grid_points, D):
    pop= initial_pop(popsize, n, grid_points, D)
    best_fit = 0
    best_layout = None
    saved_layout ={}
    for g in range(gen): #fitness
        scores = [fitness(layout) for layout in pop]
        best_index= np.argmax(scores)
        if scores[best_index] > best_fit:
            best_fit = scores[best_index]
            best_layout = pop[best_index]
            print(f"Gen {g+1}: Best Power = {scores[best_index]/1e6:.2f} MW
")
        if g in [0,9,49]:
            saved_layout[g] = pop[best_index]
        #selection = top 25%
        sorted_pop = [x for _, x in sorted(zip(scores,pop), key = lambda
pair: pair[0], reverse = True)]
        top = sorted_pop[:popsize // 4]

        #make new population
        new_pop = top.copy()
        while len(new_pop) < popsize:
            index = np.random.choice(len(top),2) #choose random parents
from top
            f1 = top[index[0]]
            f2 = top[index[1]]
            offspring = crossover(f1,f2, grid_points)
            offspring = mutate(offspring, rate, grid_points)
            new_pop.append(offspring)

        pop = new_pop

    return best_layout, best_fit, saved_layout




bestlayout, bestfitness, saved_layout = genetic_algorithm(gen, popsize,
    rate, n, grid_points, D)
print(f" Final Best Power Output: {bestfitness/1e6:.2f} MW")
inc = incident_velocities(bestlayout)
print(inc)


#%%
x, y = bestlayout[:, 0], bestlayout[:, 1]

def plot_layout(layout, title, width, height):
    x, y = layout[:, 0], layout[:, 1]
    plt.figure(figsize=(8, 8))
    plt.scatter(x, y, c='blue', s=100, edgecolors='black', marker='<',
    label='Makani M600s')
```

```
225
226     for i in range(len(layout)):
227         plt.arrow(x[i], y[i], 2*D, 0, head_width=50, head_length=100, fc='
        red', ec='red', alpha=0.3)
228
229     plt.xlim(-500, width + 500)
230     plt.ylim(-500, height + 500)
231     plt.gca().set_aspect('equal')
232     plt.grid(True)
233     plt.xlabel("x-position (m)")
234     plt.ylabel("y-position (m)")
235     plt.title(title)
236     plt.legend()
237     plt.tight_layout()
238     plt.show()
239
240 plot_layout(saved_layout[0], "Layout at Generation 1", width, height)
241 plot_layout(saved_layout[9], "Layout at Generation 10", width, height)
242 plot_layout(saved_layout[49], "Layout at Generation 50", width, height)
243 plot_layout(bestlayout, "Final Optimized Layout", width, height)
```