ΤΕΙ ΠΕΙΡΑΙΑ

ΣΧΟΛΗ ΤΕΧΝΟΛΟΓΙΚΩΝ ΕΦΑΡΜΟΓΩΝ

ΤΜΗΜΑ ΜΗΧΑΝΟΛΟΓΙΑΣ

ΕΡΓΑΣΤΗΡΙΟ Η.Μ.Ε & ΠΡΟ.ΠΕ.

ΠΤΥΧΙΑΚΗ ΕΡΓΑΣΙΑ

ΘΕΜΑ: "ΜΕΛΕΤΗ ΑΥΤΟΝΟΜΟΥ ΕΝΕΡΓΕΙΑΚΟΥ ΥΒΡΙΔΙΚΟΥ ΣΥΣΤΗΜΑΤΟΣ ΜΕ ΤΗ ΣΥΜΜΕΤΟΧΗ ΑΙΟΛΙΚΟΥ ΠΑΡΚΟΥ ΚΑΙ ΣΥΣΤΗΜΑΤΟΣ ΑΠΟΘΗΚΕΥΣΗΣ ΕΝΕΡΓΕΙΑΣ ΣΥΜΠΙΕΣΜΕΝΟΥ ΑΕΡΑ (CAES), ΜΕΣΩ ΑΝΑΠΤΥΞΗΣ ΛΟΓΙΣΜΙΚΟΥ ΠΡΟΣΟΜΟΙΩΣΗΣ"

ΣΠΟΥΔΑΣΤΕΣ : ΜΑΡΚΟΓΙΑΝΝΑΚΗΣ ΑΧΙΛΛΕΑΣ, ΠΑΣΑΣ ΠΕΤΡΟΣ

ΕΠΙΒΛΕΠΩΝ ΚΑΘΗΓΗΤΗΣ : Δρ ΙΩΑΝΝΗΣ ΚΑΛΔΕΛΛΗΣ

ΠΕΙΡΑΙΑΣ 2012

Περίληψη

Η ηλεκτροδότηση αυτόνομων δικτύων νησιωτικών περιοχών στο Αιγαίο Πέλαγος, παρέχεται έως σήμερα με την χρήση τοπικών θερμικών σταθμών παραγωγής οι οποίοι αυξάνουν σημαντικά το κόστος παραγωγής και επιβαρύνουν το περιβάλλον. Από την άλλη πλευρά, η φύση των δικτύων μεταφοράς, η μειωμένη ευελιξία των εγκατεστημένων θερμικών μονάδων αλλά και η ανακολουθία παραγωγής-ζήτησης καθιστούν αδύνατη την εκτεταμένη διείσδυση των ανανεώσιμων πηγών ενέργειας (ειδικά της αιολικής ενέργειας). Συνέπεια των παραπάνω είναι η αυξημένη απόρριψη της ενέργειας που παράγεται από αιολικά πάρκα.[17]

Η ανάπτυξη τεχνολογιών αποθήκευσης ενέργειας μπορεί να συνεισφέρει στη λύση του παραπάνω προβλήματος. Τα οφέλη από την υιοθέτηση τεχνολογιών αποθήκευσης ενέργειας είναι πολλαπλά, μιας και δίνουν την δυνατότητα εκμετάλλευσης της απορριπτόμενης από το δίκτυο ενέργειας των αιολικών πάρκων, ενισχύουν την αξιοπιστία του δικτύου διανομής και δίνουν την δυνατότητα βελτιστοποίησης της λειτουργίας των υφιστάμενων συμβατικών μονάδων παραγωγής.[16] Στον αντίποδα, οι τεχνολογίες αποθήκευσης αντιμετωπίζονται με σκεπτικισμό, λόγω του υψηλού κόστους της αρχικής επένδυσης και των απωλειών κατά την μετατροπή της ενέργειας.[3]

Πάνω σε αυτό το πλαίσιο μελετάται η υιοθέτηση συστήματος αποθήκευσης ενέργειας συμπιεσμένου αέρα, το οποίο εκμεταλλεύεται τις απορρίψεις ενός αιολικού πάρκου, για την κάλυψη των ενεργειακών αναγκών αυτόνομων νησιωτικών δικτύων. [17,18]Για την εξασφάλιση της αδιάλειπτης παροχής ενέργειας προς το δίκτυο, το σύστημα έχει την δυνατότητα να λειτουργεί και σε συμβατικό κύκλο αεριοστροβίλου με την χρήση φυσικού αερίου ως καύσιμο.[8]

Abstract

The electrification of autonomous electrical networks existing in the Aegean Archipelago Islands is principally undertaken by local, oil-dependent thermal power stations, responsible for considerable environmental impact and rather high electricity generation cost. On the other hand, the extensive exploitation of RES (especially wind energy) in autonomous (in most cases weak) electrical networks, is subject to strict constraints and barriers owed to the electricity networks nature, the operational features of the employed diesel/heavy oil units and the disharmony noted between the instantaneous wind energy production and the corresponding electricity demand of the local network. As a result, substantial amounts of wind energy being rejected by the local electricity grid may be encountered. [17]

Meanwhile, the development of energy storage systems suggests an appreciable solution that may compensate for the encountered wind energy rejections. The benefits stemming from the adoption of energy storage systems as ancillary power units are the exploitation of otherwise wasted amounts of energy (e.g. rejected amounts of wind energy can be stored), the increased reliability of energy supply (since an extra power source is available) and the improved operation of already existing power units (e.g. operation of conventional units at optimum point). On the other hand, storage technologies are faced with some scepticism arising from the high initial cost of the system and the inherent transformation losses. [3]

In this context, the adoption of a Wind-Compressed Air Energy Storage (Wind-CAES) configuration will be investigated. To ensure that the system will be able to fulfil the peak demand energy requirements even in the case that the energy stored is inadequate, the CAES unit will be able to shift its mode to the typical gas turbine cycle as well (i.e. the gas turbine will be coupled to the compressor via the use of a clutch) and use natural gas instead of wind power. [8]

Περιεχόμενα

Διαγράμματα

Πρόλογος

Θα θέλαμε να ευχαριστήσουμε θερμά τον επιβλέποντα καθηγητή μας Δρ. Ιωάννη Καλδέλλη για την υποστήριξή του, τον καθηγητή μας κ. Δημήτρη Ζαφειράκη για τις συμβουλές του και την συνεχή καθοδήγηση πάνω στην εργασία μας. Επίσης θα θέλαμε να εκφράσουμε την ευγνωμοσύνη μας στον φίλο μας Αλέξανδρο Σιγαρά για την υποστήριξή του στην δημιουργία του λογισμικού.

1. Αποθήκευση ενέργειας

1.1 *Εισαγωγή-Αποθήκευση ενέργειας σε μεγάλη κλίμακα*

Τα τελευταία χρόνια όλο και περισσότερες μέθοδοι και τεχνικές αναπτύσσονται στον τομέα της αποθήκευσης της ενέργειας. Οι δύο βασικοί λόγοι για την δημιουργία διατάξεων αποθήκευσης ενέργειας είναι:

- Η εξισορρόπηση του δικτύου
- Η ανάκτηση και επαναχρησιμοποίηση της ενέργειας σε στιγμές αυξημένης ζήτησης

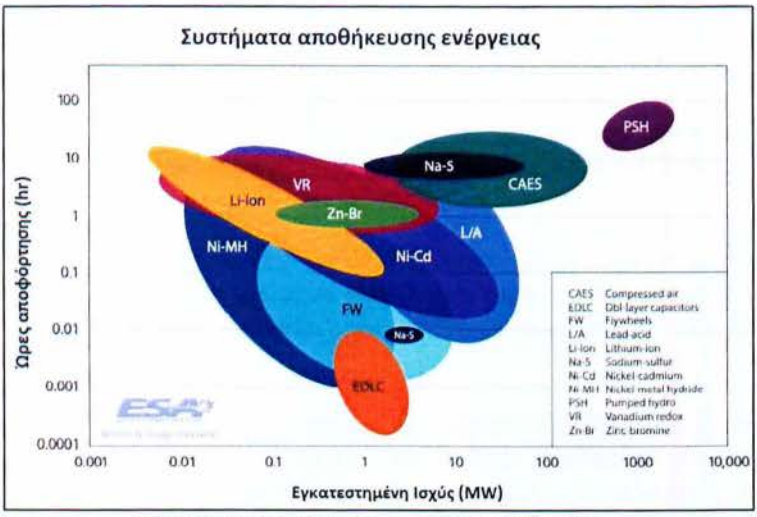
Στην πρώτη περίπτωση η αποθήκευση ενέργειας βοηθάει την διατήρηση της σταθερότητας του δικτύου για μικρά χρονικά διαστήματα, ενώ στην δεύτερη συνεισφέρει στην κάλυψη της ζήτησης σε ώρες αιχμής όπου οι μονάδες βάσης δεν μπορούν να ανταποκριθούν.[1]

Σχετικά με την χρήση Ανανεώσιμων Πηγών Ενέργειας (ΑΠΕ), παρατηρείται ασυμφωνία μεταξύ παραγωγής και ζήτησης της ενέργειας. Αυτό το χάσμα καλούνται να γεφυρώσουν τα συστήματα αποθήκευσης ενέργειας, τα οποία μπορούν να εξασφαλίσουν την μέγιστη αξιοποίηση της παραγόμενης ενέργειας με το λιγότερο δυνατό οικονομικό και περιβαλλοντικό κόστος σε σύγκριση με την αύξηση των μονάδων παραγωγής. Ακόμη, η βέλτιστη διαχείριση ενέργειας και η ορθολογική χρήση των διαθέσιμων πηγών, σε συνδυασμό με την ενίσχυση της ενεργειακής αυτονομίας σε μη διασυνδεδεμένα δίκτυα της ηπειρωτικής και της νησιωτικής χώρας, ενθαρρύνουν την χρήση μεθόδων αποθήκευσης. [7]

Όπως προαναφέρθηκε, έχουν αναπτυχθεί αρκετές μέθοδοι αποθήκευσης της ενέργειας, οι σημαντικότερες από τις οποίες περιγράφονται παρακάτω:

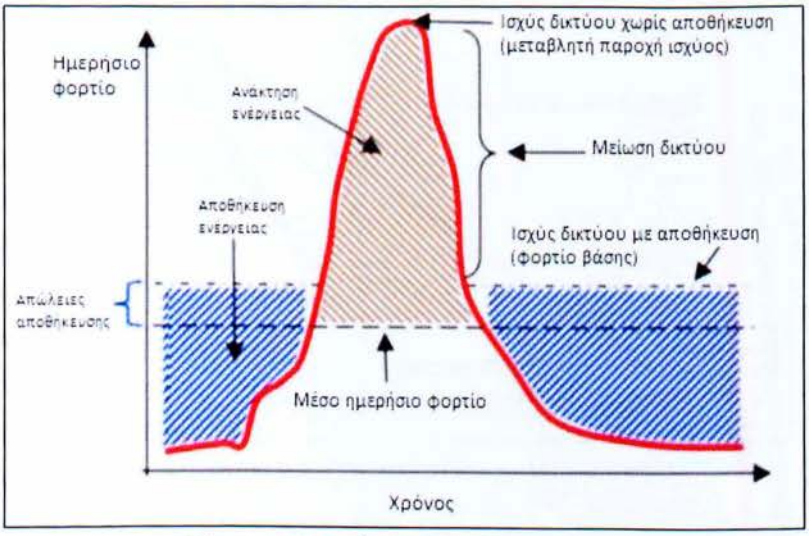
- Αναστρέψιμα υδροηλεκτρικά (Pumped Hydro)
- Αποθήκευση συμπιεσμένου αέρα (CAES)
- Σφόνδυλοι (Flywheels)
- Αποθήκευση σε μαγνητικό πεδίο (SMES)
- Υπερπυκνωτές (Super Capacitors)
- Συσσωρευτές (Batteries)
- Στοιχεία ροής (Flow Batteries)

Στο παρακάτω διάγραμμα παραθέτονται οι τεχνολογίες αποθήκευσης αναλόγως με την δυνατότητα παροχής ενέργειας στο δίκτυο.



Διάγραμμα 1: Τεχνολογίες αποθήκευσης ενέργειας

Σκοπός των τεχνολογιών αποθήκευσης ενέργειας είναι αφενός η ανάκτηση της μη εκμεταλλεύσιμης παραγόμενης ενέργειας και η χρήση της σε περιόδους αιχμής [24] και υψηλής ζήτησης, όπου οι μονάδες βάσης δεν μπορούν να καλύψουν το φορτίο και αφετέρου η ενίσχυση της σταθερότητας του δικτύου[4].



Διάγραμμα 2: Διακύμανση φορτίου στο δίκτυο

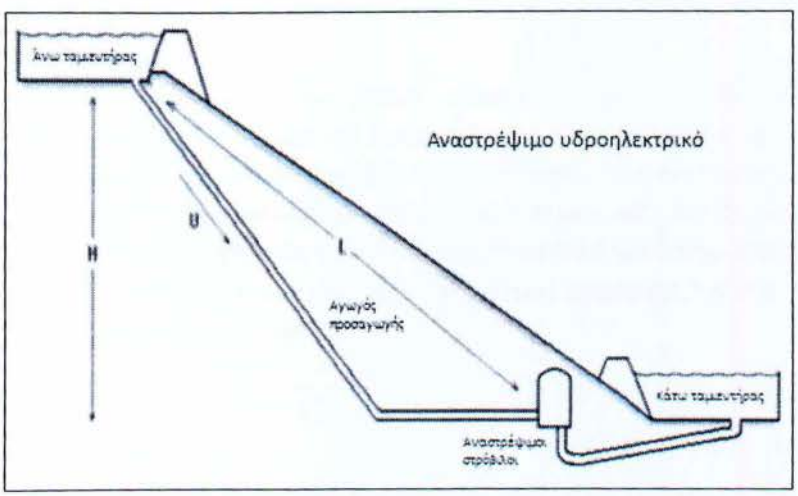
1.1.1 *Αναστρέψιμα υδροηλεκτρικά*

Οι αναστρέψιμοι υδροηλεκτρικοί σταθμοί δίνουν την δυνατότητα παραγωγής ενέργειας αλλά και αποθήκευσής της μέσω αντίστροφης λειτουργίας. Τα αναστρέψιμα υδροηλεκτρικά διαθέτουν δύο ταμιευτήρες νερού, έναν σε μεγάλο υψόμετρο και έναν στην βάση. Κατά την συμβατική λειτουργία η ροή του νερού δίνει κίνηση σε υδροδυναμικές μηχανές (στροβίλους) οι οποίες παράγουν ηλεκτρική ενέργεια. Όταν υπάρχει περίσσεια ενέργειας στο δίκτυο, χρησιμοποιούνται αντλίες οι οποίες μεταφέρουν το νερό στον άνω ταμιευτήρα.[10]

Σημαντικό πλεονέκτημα των υδροηλεκτρικών σταθμών παραγωγής είναι η δυνατότητα γρήγορης παραλαβής και απόρριψης φορτίου μεγάλης ισχύος, γεγονός που τους καθιστά ικανούς για την κάλυψη των αιχμών φορτίου που παρουσιάζονται στο δίκτυο.[9]

Η πρώτη εφαρμογή μεγάλης κλίμακας αναφέρεται το 1929 στην Γερμανία. Τα αναστρέψιμα υδροηλεκτρικά έργα που λειτουργούν σήμερα σε ολόκληρη την υφήλιο έχουν ισχύ περί τις 140 GW από τα οποία: 100 GW στην Ευρώπη, την Ασία και την Λατινική Αμερική, 21 GW στην Ιαπωνία και 19 GW στις ΗΠΑ. Στην Ελλάδα λειτουργούν 2 αναστρέψιμα υδροηλεκτρικά έργα: [10]

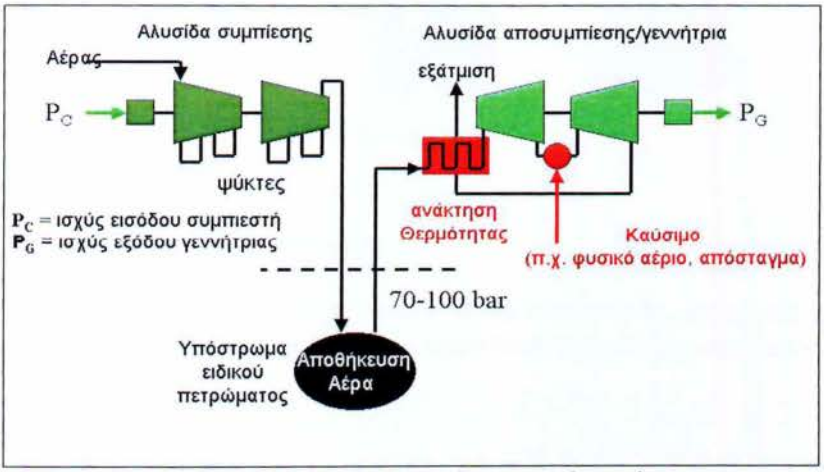
- Της Σφηκιάς στον ποταμό Αλιάκμονα (1985) με 3 αναστρέψιμες μονάδες ισχύος 3x105=315 MW.
- Του Θησαυρού στον ποταμό Νέστο (1998) με 3 αναστρέψιμες μονάδες ισχύος 3x127=381 MW, διαθέσιμη υδραυλική πτώση H=154 m και ταμιευτήρα χωρητικότητας 565.106 m^3



Εικόνα 1: Διάταξη αναστρέψιμου υδροηλεκτρικού

1.1.2 Αποθήκευση συμπιεσμένου αέρα

Με την μέθοδο αποθήκευσης ενέργειας με την μορφή συμπιεσμένου αέρα δίνεται η δυνατότητα κάλυψης ενεργειακών αναγκών σε περιόδους αιχμής. Ο συμπιεσμένος αέρας φυλάσσεται σε φυσικά αεροθυλάκια ή δοχεία πίεσης. [22]Στην συνέχεια, αναμειγνυόμενος με καύσιμο καίγεται παράγοντας καυσαέρια τα οποία εκτονώνονται σε αεριοστρόβιλο δίνοντας κίνηση στη γεννήτρια που παράγει ρεύμα.[20,23]



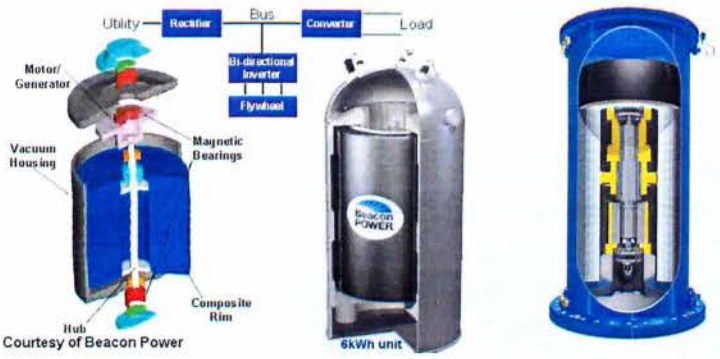
Εικόνα 2: Τυπική διάταξη συστήματος συμπίεσης αέρα

Μια βασική παράμετρος που χρήζει προσοχής είναι η ρύθμιση της θερμοκρασίας αποθήκευσης. Κατά τη διαδικασία της συμπίεσης του αέρα ένα μέρος της ενέργειας χάνεται με τη μορφή θερμότητας, γεγονός μη επιθυμητό. Εκτός αυτού, κατά την εκτόνωση απελευθερώνονται αρκετά σημαντικά ποσά θερμότητας τα οποία αν αξιοποιηθούν μπορούν να βελτιώσουν σημαντικά τον βαθμό απόδοσης της μονάδας. Τα τελευταία χρόνια γίνεται μια σημαντική προσπάθεια αξιοποίησης αυτών μέσω των αδιαβατικών CAES.[13]

1.1.3 Σφόνδυλοι

Ο σφόνδυλος αποτελείται από ένα συμπαγή δίσκο που συγκρατείται μέσω μαγνητικών εδράνων και είναι προσαρμοσμένος σε έναν άξονα περιστροφής. Όσο ταχύτερα περιστρέφεται ο σφόνδυλος τόσο περισσότερη ενέργεια παράγεται, η

οποία αποδίδεται χρησιμοποιώντας τη ροπή του σφονδύλου διεγείροντας την ηλεκτρογεννήτρια. [5]



Εικόνα 3: Σφόνδυλοι

1.1.4 Συσσωρευτές-Μπαταρίες

Οι μπαταρίες αποτελούν τον πλέον γνωστό τρόπο αποθήκευσης της ηλεκτρικής ενέργειας υπό τη μορφή χημικής ενέργειας. Οι πιο διαδεδομένες είναι οι εξής: [2]

- Οξέων μολύβδου
- Νικελίου-Καδμίου
- Θείου-Νατρίου
- Χλωριδίου νικελίου-Νατρίου
- Ιόντων Λιθίου
- Μεταλλικού στοιχείου-Αέρα

Εικόνα 4: Συστοιχία μπαταριών

1.1.5 Μαγνητικά Πεδία

Η αποθήκευση ενέργειας υπό μορφή μαγνητικών πεδίων επιτυγχάνεται με την ροή ρεύματος σε υπεραγώγιμο πηνίο σε αρκετά χαμηλές θερμοκρασίες (55-77 Kelvin). [15]

1.1.6 Υπερπυκνωτές

Η αρχή λειτουργίας των υπερπυκνωτών είναι αντίστοιχη των πυκνωτών με τη διαφορά ότι έχουν τη δυνατότητα να αποθηκεύουν περισσότερη ενέργεια μέσω πολυμερών, επιτυγχάνοντας χιλιάδες ή ακόμη και εκατομμύρια φορές μεγαλύτερη ενεργειακή πυκνότητα από τους συμβατικούς πυκνωτές (5F/cm^2 αντί για 40μF/cm^2) έχοντας ευρεία εφαρμογή στα υβριδικά αυτοκίνητα (τεχνολογία boost). [15]

Εικόνα 5: Υπερπυκνωτής

Αποθηκευτική Διάταξη	Πλεονεκτήματα	Μειονεκτήματα	Συνήθης εφαρμογή
Μπαταρίες ροής (flow): PSB, VRBr, ZnBr	Υψηλή χωρητικότητα, ανεξάρτητη εκτίμηση ισχύος - ενέργειας	Χαμηλή πυκνότητα ενέργειας	Εξομάλυνση ζήτησης λίγων ωρών
Μολύβδου – οξέος	Χαμηλό αρχικό κόστος	Περιορισμένος κύκλος ζωής σε βαθιά εκφόρτιση	Εξομάλυνση αιχμών
Ni – Cd	Υψηλή πυκνότητα ενέργειας και ισχύος, απόδοση		Εξομάλυνση ζήτησης λίγων ωρών-λεπτών
Li – ion	Υψηλή πυκνότητα ισχύος και ενέργειας, υψηλή απόδοση	Υψηλό κόστος παραγωγής, απαιτεί ειδικό κύκλωμα φόρτισης	Κινητή τηλεφωνία, υποσταθμοί ενέργειας
NaS	Υψηλή πυκνότητα ισχύος και ενέργειας, υψηλή απόδοση	Κόστος παραγωγής, μέτρα ασφαλείας (λόγω σχεδιασμού)	Εξομάλυνση ζήτησης λίγων ωρών-λεπτών
Σφόνδυλοι (flywheels)	Υψηλή ισχύς	Χαμηλή πυκνότητα ενέργειας	Εξομάλυνση ισχύος για λίγα λεπτά
SMES (Υπεραγώγιμη Μαγνητική Αποθήκευση),	Υψηλή ισχύς	Χαμηλή πυκνότητα ενέργειας, υψηλό κόστος παραγωγής	Εφαρμογές ποιότητας ισχύος, διανομή
E.C Capacitors[22]	Μεγάλος κύκλος ζωής, υψηλή απόδοση	Χαμηλή πυκνότητα ενέργειας	Εφαρμογές ποιότητας ισχύος, διανομή
Αντλησιοταμίευση (pumped storage)	Υψηλή χωρητικότητα, χαμηλό κόστος	Απαιτεί ειδική τοποθεσία	Εξομάλυνση ζήτησης σε μεγάλο χρονικό διάστημα
Ενεργειακή Αποθήκευση Συμπιεσμένου αέρα CAES	Υψηλή χωρητικότητα, χαμηλό κόστος	Απαιτεί ειδική τοποθεσία για τις χρησιμοποιούμενες κοιλότητες	Εξομάλυνση ζήτησης σε μεγάλο χρονικό διάστημα

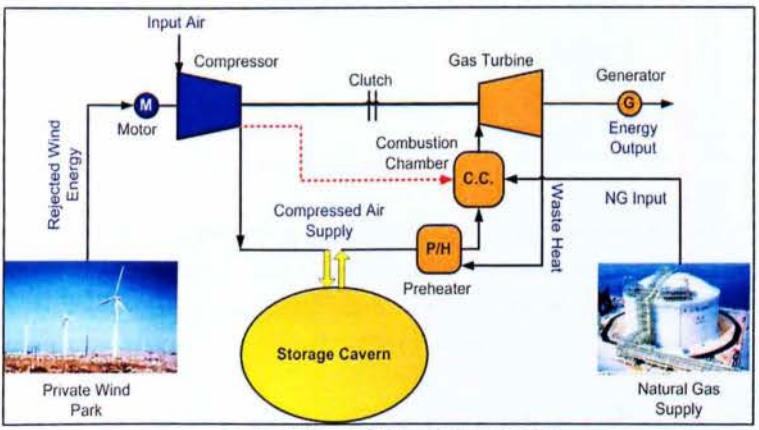
Πίνακας 1. Συγκριτικά χαρακτηριστικά μεθόδων αποθήκευσης

1.2 Αποθήκευση απορριπτόμενης ενέργειας ανεμογεννητριών μέσω συμπίεσης αέρα (Wind-CAES)

1.2.1 Περιγραφή

Οι ανεμογεννήτριες παράγουν ηλεκτρική ενέργεια ανεξαρτήτως της ζήτησής της από το δίκτυο. Μέσω των συστημάτων αποθήκευσης συμπιεσμένου αέρα, η παραγόμενη ενέργεια που δεν είναι δυνατό να απορροφηθεί από το δίκτυο μετατρέπεται σε άλλη μορφή ενέργειας ώστε να αποθηκευτεί. [25,26] Συγκεκριμένα, στο σύστημα αποθήκευσης συμπιεσμένου αέρα (Compressed Air Energy Storage) διοχετεύεται η περίσσεια ενέργειας προς συμπιεστή ώστε να αποθηκευτεί σε αεροθυλάκιο ή δοχεία πίεσης. Όταν υπάρξει έλλειψη σε ηλεκτρική ενέργεια χρησιμοποιείται συμπιεσμένος αέρας, ώστε μέσω της διάταξης του κύκλου Brayton να καλυφθεί η ζήτηση ενέργειας.

Μια γενική περίπτωση του CAES είναι η ανάκτηση ενέργειας κατά την διάρκεια της νύχτας, και η χρήση της κατά τις ώρες αιχμής όπου οι μονάδες βάσης δεν μπορούν να καλύψουν την ζήτηση. [19]Τα αιολικά πάρκα, παράγουν ενέργεια και τις νυχτερινές ώρες, λόγω όμως της χαμηλής ζήτησης και της υπερκάλυψης του φορτίου από τις μονάδες βάσης δεν είναι δυνατόν αυτή να απορροφηθεί από το δίκτυο. Με τον τρόπο αυτό, όπως και στα αναστρέψιμα υδροηλεκτρικά χρησιμοποιείται η περίσσια για την κάλυψη της ζήτησης σε ώρες αιχμής.



Εικόνα 6: Τυπική διάταξη Wind-CAES

Κατά την διάρκεια της ημέρας όπου το ζητούμενο φορτίο είναι αυξημένο διοχετεύεται ο συμπιεσμένος αέρας απευθείας στον αεριοστρόβιλο του κύκλου Brayton εξοικονομώντας μεγάλη ποσότητα καυσίμου από την διαδικασία συμπίεσης.

1.2.2 Αρχή λειτουργίας

Εκμεταλλευόμενοι την περίσσεια ενέργειας από το αιολικό πάρκο κατά τις περιόδους χαμηλής ζήτησης, είμαστε σε θέση να την ανακτήσουμε κατά τις περιόδους αιχμής για την κάλυψη της ζήτησης.

Μεγάλη ποσότητα αέρα συμπιέζεται με σκοπό την αποθήκευση του σε υπόγειους θύλακες ή δοχεία πίεσης υπό υψηλή πίεση. Στην συνέχεια, ο συμπιεσμένος αέρας οδηγείται στο θάλαμο καύσης, αναμιγνύεται με την απαιτούμενη ποσότητα καυσίμου (στην προκειμένη περίπτωση φυσικό αέριο) και αναφλέγεται. Έτσι, η ενέργεια των καυσαερίων που παράγονται περιστρέφει τον αεριοστρόβιλο και μέσω ηλεκτρογεννήτριας μετατρέπεται το μηχανικό έργο σε ηλεκτρισμό. Στην περίπτωση που η ποσότητα του αέρα δεν επαρκεί για να καλύψουμε την ζητούμενη ενέργεια, συμπλέκεται ο συμπιεστής με το στρόβιλο και λειτουργεί με το συμβατικό κύκλο Brayton.

Τα βασικότερα μέρη της μονάδας είναι τα ακόλουθα (βλέπε Εικόνα 6):

- Ο ηλεκτροκινητήρας που κινεί τον συμπιεστή
- Ο συμπιεστής
- Το αεροθυλάκιο
- Ο συμπλέκτης
- Ο στρόβιλος
- Η γεννήτρια

1.2.3 Υφιστάμενες εγκαταστάσεις

1.2.3.1 Η μονάδα του **Huntorf,** Γερμανία

Η μονάδα του Huntorf, βρίσκεται στη Βόρεια Γερμανία, κατασκευάστηκε το 1978 και είναι η πρώτη εγκατάσταση αποθήκευσης συμπιεσμένου αέρα στον κόσμο. Σκοπός της κατασκευής αυτού του έργου είναι η άμεση παραγωγή ηλεκτρικής ενέργειας για την κάλυψη της αυξημένης ζήτησης.[6]

Γενικός Σχεδιασμός

Ο συνολικός όγκος των περίπου 310,000m^3 αεροθυλακίου ανταποκρίνεται στις πιέσεις λειτουργίας της μονάδας και την απαιτούμενη ποσότητα αέρα. Μεγάλη έμφαση δόθηκε στα θερμοδυναμικά χαρακτηριστικά του αεροθυλακίου κατά τη διάρκεια της αποθήκευσης και της ανάκτησης. Θα μπορούσε να είχε κατασκευαστεί ένα μόνο θυλάκιο αλλά προτιμήθηκε η κατασκευή δυο επιμέρους για τους παρακάτω λόγους:[12]

- Καλύτερη διαχείριση στο άνοιγμα και το κλείσιμο του αεροθαλάμου.
- Ευκολότερη προσαρμογή του αεροθαλάμου σε περιπτώσεις πλήρους αποσυμπίεσης και επαναφοράς.
- Κατά την εκκίνηση απαιτείται ελάχιστη πίεση των 13 bar σε κάθε ένα από τα δυο αεροθυλάκια.

Το βάθος των δεξαμενών έχει σχεδιαστεί έτσι ώστε να διασφαλίζει τη σταθερότητα για αρκετούς μήνες σε συνθήκες ατμοσφαιρικής πίεσης, και να εξασφαλίζει την μέγιστη πίεση των 100 bar. Και τα δύο αεροθυλάκια αντέχουν εξαιρετικά μεγάλες παροχές αέρα της τάξης των 417kg/sec επιτυγχάνοντας πολύ μικρές απώλειες πίεσης.

Παρακάτω αναφέρονται τα χαρακτηριστικά της μονάδας.[12]

Ισχύς	
Στροβίλου	*290 MW (< 3 hrs)*
Συμπιεστή	*60 MW (< 12 hrs)*
Παροχή αέρα	
Στρόβιλος	*417 kg/s*
Συμπιεστής	*108 kg/s*
Λόγος εισόδου/ εξόδου	1/4
Αριθμός αεροθυλακίων	*2*
Όγκος αεροθαλάμου (1)	*~140 000 m^3*
Όγκος αεροθαλάμου (2)	*~170 000 m^3*
Συνολικός όγκος	*~310 000 m^3*
Διαστάσεις θυλακίου	
Κορυφή	*~650 m*
Κάτω μέρος	*~800 m*
Μέγιστη διατομή	*~ 60m*
Well spacing	*220m*
Πιέσεις αεροθυλακίων	
Ελάχιστη επιτρεπόμενη	*1 bar*
Ελάχιστη λειτουργική	*20 bar*
Ελάχιστη λειτουργική (συνηθισμένη)	*43 bar*
Ελάχιστη συνολική λειτουργική	*70 bar*
Μέγιστος ρυθμός εκφόρτισης	*15 bar/h*

Πίνακας 2: Προδιαγραφές της μονάδας Huntorf

Λειτουργία και τρόπος διαχείρισης

Η διαδικασία της εκκίνησης του συμπιεστή απαιτεί μια ελάχιστη πίεση 13bar, η οποία επιτυγχάνεται μέσω μερικής πλήρωσης του ενός αεροθυλακίου από το γειτονικό του. Ο αριθμός των εκκινήσεων που γίνονται έχει μια ευρεία διακύμανση που αποδίδεται στα παρακάτω:

- Σύνδεση με ένα μεγαλύτερο δίκτυο από το 1985 το οποίο προστέθηκε στην δίκτυο των αναστρέψιμων υδροηλεκτρικών σταθμών.
- Ο αρχικός ρόλος του CAES είναι να λειτουργεί ως μια εφεδρική μονάδα σε περίπτωση βλάβης των κύριων μονάδων ηλεκτροπαραγωγής.
- Λειτουργία ως εναλλακτική επιλογή στην κάλυψη των υψηλών σημείων ζήτησης.

Η μονάδα χρησιμοποιείται για την άμεση κάλυψη ενέργειας για τα μεσαίου μεγέθους συμβατικά ηλεκτροπαραγωγά ζεύγη. Χρειάζονται περίπου 3-4 ώρες ώστε να τεθεί σε πλήρη ισχύ αφού πρώτα είναι σε θέση να παράγει μικρή αλλά σταθερή ποσότητα ενέργειας. Επιπρόσθετα, ακόμη μια χρήση της παρούσας μονάδας είναι να αποθηκεύει την περίσσεια ενέργειας που παράγεται την νύχτα ώστε να καλύψει τα "peaks" της επόμενης ημέρας όταν δεν είναι σε θέση να αποθηκευτεί άλλη ενέργεια στα αναστρέψιμα υδροηλεκτρικά. Τέλος, η σημαντική αύξηση του αιολικού δυναμικού της Βόρειας Γερμανίας ενίσχυσε και ενισχύει το ενδιαφέρον περαιτέρω εκμετάλλευσης της μονάδας αυτής στην περιοχή.[12]

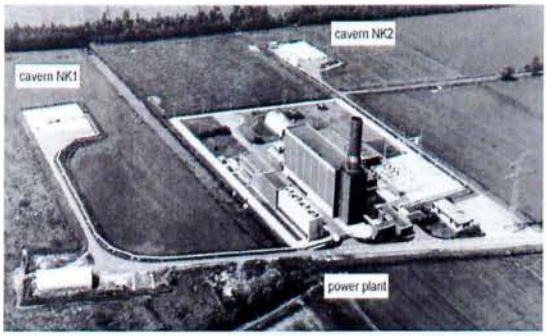
Θερμοδυναμικά χαρακτηριστικά

Κατά τη διάρκεια του πρώτου γεμίσματος και των απαιτούμενων ελέγχων, πραγματοποιήθηκαν εκτεταμένες μετρήσεις πιέσεων και θερμοκρασιών. Η προσομοίωση αποκάλυψε ότι η μετάδοση θερμότητας με το πετρώδες τοίχωμα λαμβάνει χώρα σε μια περιφερειακή ζώνη πάχους ενός μέτρου. Τα ακανόνιστα σχήματα των αεροθυλακίων έχουν το πλεονέκτημα να αυξάνουν την μετάδοση θερμότητας μεταξύ του αέρα και των τοιχωμάτων της δεξαμενής, το οποίο οδηγεί σε σημαντική αύξηση της χωρητικότητας. Όταν ο αέρας εκτονώνεται στην ατμοσφαιρική πίεση , η πίεση μειώνεται σημαντικά με αποτέλεσμα την μείωση της θερμοκρασίας του αέρα, αλλά όταν φτάσει στο ελάχιστο επιτρεπτό όριο η θερμοκρασία αυξάνεται και πάλι.[12]

Συντήρηση της μονάδας

Η αξιολόγηση των τοιχωμάτων των αεροθυλακίων αποδεικνύεται δύσκολη μέσω των συνηθισμένων μεθόδων γιατί τα όργανα μέτρησης είναι ευαίσθητα στην υγρασία του συμπιεσμένου αέρα και συν τοις άλλοις τα ευρέως διαδεδομένα lasers είναι αναποτελεσματικά λόγω της ομίχλης που επικρατεί στο εσωτερικό. Όταν το

θυλάκιο NK1 εκτονώθηκε στη θερμοκρασία περιβάλλοντος στις αρχές του 2001, η πιο αποτελεσματική μέθοδος ήταν η χρήση θερμαινόμενου laser. Τα αποτελέσματα αυτής της αποτίμησης μετά από 20 χρόνια, δεν έδειξαν πρακτικά καμία απόκλιση από τις προβλεπόμενες και τις πραγματικές συνθήκες. Μόνο μετά από μερικούς μήνες λειτουργίας, εντοπίστηκαν σοβαρά σημάδια διάβρωσης στη διάταξη του στροβίλου, με αποτέλεσμα να απενεργοποιείται απρογραμμάτιστα η μονάδα μετά από την εκκίνηση του σταθμού.[12]



Εικόνα 7: Η μονάδα του Huntorf στη Γερμανία

1.2.3.2 Η μονάδα του McIntosh, Alabama Η.Π.Α

Γενικός Σχεδιασμός

Από το 1991 μια μονάδα συμπίεσης αέρα 110MW στο McIntosh της Αλαμπάμα κατασκευάστηκε για να παράγει ενέργεια κατά τις περιόδους αιχμής. Η διαφορά με την μονάδα της Γερμανίας είναι ότι εκμεταλλεύεται τις απορρίψεις ενέργειας των συμβατικών μεθόδων παραγωγής για τη συμπίεση του αέρα, έτσι παράγεται ενέργεια η οποία πωλείται στις περιόδους αιχμής. Μια καθημερινή εκκίνηση του συστήματος απαιτεί 14 λεπτά ώστε να επιτευχθεί η ισχύς των 110MW, ενώ ένας πομποδέκτης ελέγχει την συμπίεση και την παραγωγή μέσω μικροκυμάτων 90 μίλια μακριά. Η διάταξη παραγωγής έχει μήκος 43 μέτρα καθιστώντας την από τις μεγαλύτερες στον κόσμο και εξοπλίζεται σχεδόν κατ' αποκλειστικότητα από την Dresser-Rand. Ο εξοπλισμός περιλαμβάνει: [13]

- Μονοβάθμιους αεριοστρόβιλους
- Πολυβάθμιους αεριοστρόβιλους
- Γεννήτριες με κιβώτια μετάδοσης
- Αξονικούς, φυγοκεντρικούς και εμβολοφόρους συμπιεστές

Κατά την διάταξη η γεννήτρια είναι τοποθετημένη κεντρικά και συνδέεται μέσω συμπλέκτη και από τις δύο πλευρές. Από την μία πλευρά είναι τοποθετημένος ο

συμπιεστής χαμηλής-μεσαίας-υψηλής πίεσης ο οποίος συμπιέζει αέρα και τον αποθηκεύει σε υπόγειο θύλακα με πίεση έως 75 bar. Για την βελτίωση της απόδοσης του κύκλου συμπίεσης χρησιμοποιούνται πολυβάθμιοι συμπιεστές και εναλλάκτες θερμότητας (ψύκτες). [13]

Εικόνα 8: Διάταξη μονάδας

13-YEAR AVERAGE RELIABILITY			
COMPRESSION		GENERATION	
Starting	Running	Starting	Running
92.7%	99.6%	91.6%	96.7%

Πίνακας 3: Στατιστικά στοιχεία μονάδας

Πλεονεκτήματα της μονάδας

1. *Αύξηση της απόδοσης και του χρόνου ζωής*. Η εγκατάσταση του CAES παρέχει τη δυνατότητα βελτιστοποίησης του βασικού φορτίου μέσω της ελαχιστοποίησης των αποκλίσεων του μέγιστου και του ελάχιστου εμφανιζόμενου φορτίου. Με αυτό τον τρόπο επιτυγχάνεται ο καλύτερος δυνατός βαθμός απόδοσης και μεγιστοποιείται ο χρόνος ζωής. Η διαδικασία αυτή είναι πιο οικονομική γιατί αφενός απαιτεί λιγότερα καύσιμα και αφετέρου δεν χρησιμοποιεί συμβατικές λύσεις για την κάλυψη των peaks.

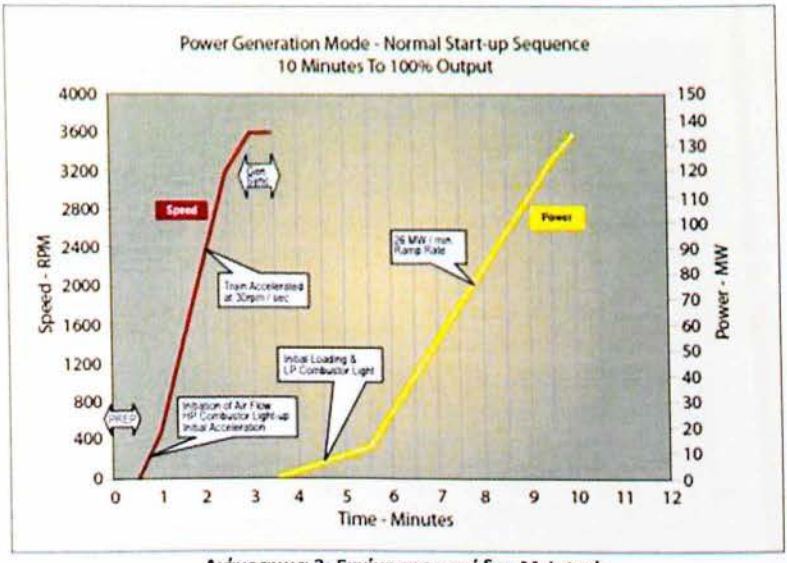
2. *Άμεση απόκριση του συστήματος.* Η γεννήτρια του CAES έχει σχεδιαστεί έτσι ώστε να φτάνει στο πλήρες φορτίο της σε μόλις 10 λεπτά. Έτσι, εξαλείφεται η ανάγκη για τη χρήση ενδιάμεσων μονάδων.

3. *Ευέλικτος κύκλος λειτουργίας.* Το σύστημα έχει την δυνατότητα να συμπιέζει αέρα όταν δεν λειτουργεί για παραγωγή ενέργειας και μπορεί να ρυθμιστεί για καθημερινούς, εβδομαδιαίους ή και εκτεταμένους κύκλους λειτουργίας. Αυτή η δυνατότητα επιτρέπει την εξισορρόπηση του δικτύου και την χρήση φθηνής ενέργειας κατά την συμπίεση.

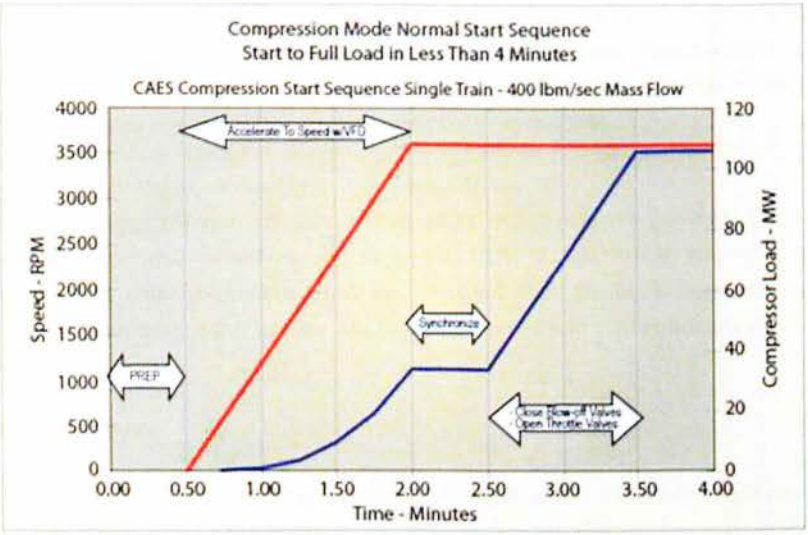
4. <u>*Φιλικό στο περιβάλλον*</u>. Η εγκατάσταση του CAES παρέχει περιβαλλοντικά οφέλη συγκριτικά με της συμβατικές μεθόδους γιατί χρησιμοποιεί το 1/3 του καύσιμου και γιατί σε αντίθεση με τις μονάδες ορυκτών καύσιμων, το CAES εκπέμπει πολύ μικρότερες ποσότητες καυσαερίων και αερίων του θερμοκηπίου. [13]

Το "έξυπνο" CAES (Smart CAES)

Το SMART-CAES έχει περισσότερα πλεονεκτήματα όσον αφορά την τεχνολογία, το περιβάλλον και την ίδια την εταιρία που το χρησιμοποιεί. Τα τεχνολογικά κατορθώματα της εγκατάστασης του McIntosh έχουν βελτιώσει την αποδιδόμενη ισχύ, την απόδοση και έχουν μειώσει την κατανάλωση του καύσιμου, του αέρα και τα επίπεδα θορύβου. Το σύστημα έχει σχεδιαστεί, όπως προαναφέρθηκε, να δίνει ισχύ 110 MW, ενώ το σύστημα ταχυτήτων παρέχει τη δυνατότητα η συμπίεση να ξεκινά σε λιγότερο από 3,5 λεπτά. Αφού συγχρονιστεί, μπορεί να αποδώσει οποιαδήποτε τιμή ισχύος μεταξύ του 15% και του 100% της ονομαστικής. Μέσα σε αυτό το εύρος λειτουργίας μπορεί να μεταβάλλει την ισχύ κατά 20% ή 27MW ανά λεπτό. Κατά την φάση της φόρτισης οι συμπιεστές μπορούν να λειτουργήσουν σε εύρος από 65 έως 100% της ονομαστικής τους ισχύος μέσω της χρήσης μεταβλητών αυλών εισαγωγής. Μέσα σε αυτό το εύρος λειτουργίας υπάρχει δυνατότητα να μεταβάλλει την ισχύ κατά 35% ανά λεπτό. [13]

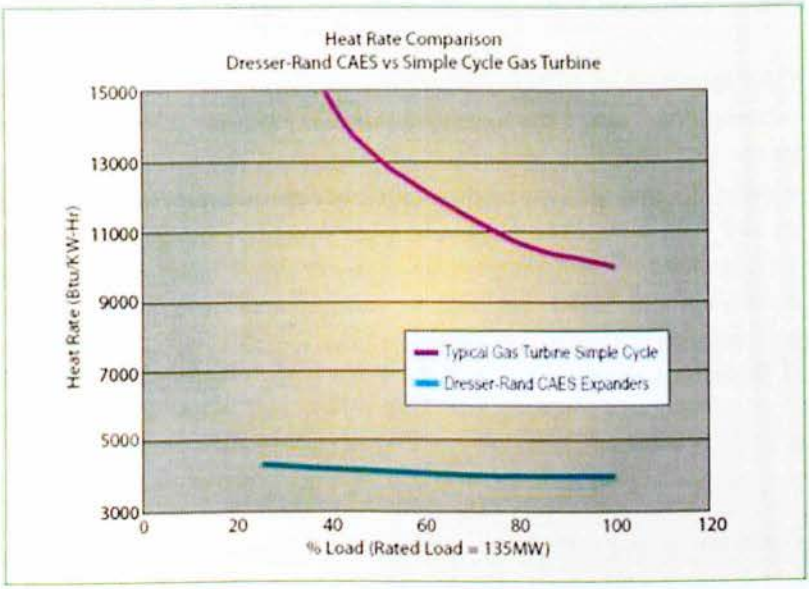


Διάγραμμα 3: Εκκίνησης μονάδας McIntosh



Διάγραμμα 4: Εκκίνηση συμπιεστή Mc Intosh

Όσον αφορά τις καταναλώσεις, η εταιρεία έχει πετύχει μια βελτίωση της τάξης του 2% στην ειδική κατανάλωση καυσίμου σε συνδυασμό με 1,2% μείωση στην ειδική κατανάλωση αέρα εντός του εύρους λειτουργίας από 20 έως 135MW. Η ειδική κατανάλωση καυσίμου των στροβίλων είναι χαμηλή και σχεδόν σταθερή για ένα εύρος λειτουργίας από 25 έως 100% του φορτίου χάρη στην ανεξάρτητη λειτουργία των στροβίλων σε σχέση με τους συμπιεστές. [13]



Διάγραμμα 5: Σύγκριση κύκλου Brayton - CAES

Ο συμπιεστής της Dresser-Rand έχει σχεδιαστεί προσεκτικά με σημαντικές βελτιώσεις στην ψύξη (intercooling) και επιτυγχάνει έναν πολυτροπικό βαθμό απόδοσης της τάξης του 80%. Επίσης, η τεχνολογία «D-R® duct resonator array» έχει πετύχει μείωση των επιπέδων του ήχου της τάξης των 10dB.

Το σύστημα ανάκτησης της θερμότητας των καυσαερίων έχει απλό σχεδιασμό και απόδοση 10% υψηλότερη από το προηγούμενο μοντέλο, φθάνοντας το υψηλό ποσοστό ανάκτησης της τάξης του 85%. Η μελετημένη χωροθέτηση των αυλών μέσα στον εναλλάκτη αλλά και το υλικό τους βοηθούν στην μείωση των επικαθήσεων και της διάβρωσης και βελτιώνουν την ροή των καυσαερίων μέσα σε αυτόν. [13]

Περιβάλλον

Οι τεχνολογικές βελτιώσεις της εγκατάστασης του smart-CAES περιλαμβάνουν και μια πρόσθετη μονάδα έλεγχου για τις εκπομπές και τη ρύθμιση των επιτρεπόμενων ορίων των NOx και CO. Μια απλή διάταξη με εγχυτήρα νερού επιτρέπει τον έλεγχο των επιπέδων των NOx και σε συνδυασμό με τον καταλύτη στο τέλος της διάταξης επιτυγχάνονται χαμηλές τελικές τιμές οξειδίων του αζώτου αλλά και ομαλή λειτουργία του συστήματος. Επιπροσθέτως, το σύστημα μειώνει τον χρόνο εκκίνησης-απόκρισης, περιορίζοντας σημαντικά τις εκπομπές στην εκκίνηση και στις περιπτώσεις που μπορεί να συνδυαστεί με ανεμογεννήτριες ή φωτοβολταϊκά οι εκπομπές των επιβλαβών αέριων μειώνονται στο ένα τρίτο. [13]

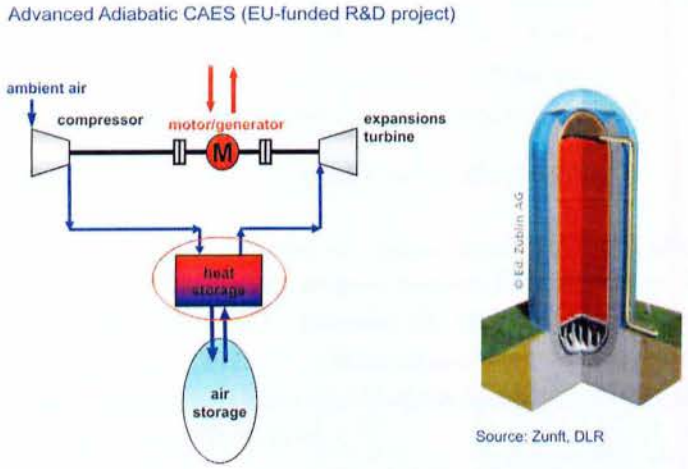
1.2.5 Αδιαβατικό CAES

Το αδιαβατικό CAES έχει την ίδια αρχή λειτουργίας με το συμβατικό CAES με τη διαφορά πως η θερμότητα του συμπιεσμένου αέρα δεν χάνεται αλλά συμπεριλαμβάνεται στη διαδικασία της παραγωγής ενέργειας. [20] Με αυτό τον τρόπο, επιτυγχάνεται αύξηση του βαθμού απόδοσης (περίπου 70%). Δεν υπάρχει κάποια υλοποιημένη εφαρμογή του αδιαβατικού CAES παρά μόνο ένα φιλόδοξο πρόγραμμα με όνομα "Adele Development Program 2010" το οποίο έχει ως σκοπό την κατασκευή ενός πειραματικού αδιαβατικού CAES. Στο πρόγραμμα αυτό συμμετέχουν οι μεγαλύτεροι κατασκευαστές εξοπλισμού παραγωγής ενέργειας όπως η General Electric και η R.W.E αλλά και η Γερμανική Διαστημική Εταιρεία DLR.[15]

Γενικός Σχεδιασμός

Η βασική αρχή λειτουργίας του αδιαβατικού CAES είναι η αποθήκευση και ανάκτηση της θερμότητας από τον συμπιεσμένο αέρα. Αυτό επιτυγχάνεται μέσω της διέλευσης του θερμού συμπιεσμένου αέρα (600°C) μέσα από μεγάλα μονωμένα

δοχεία πίεσης τα οποία περιέχουν κεραμικά και άλλα υλικά στα οποία αποθηκεύεται η θερμότητα του αέρα. Στην συνέχεια, ο αέρας αποθηκεύεται σε υπόγειες κοιλότητες. Κατά την αντίστροφη λειτουργία, ο αέρας διοχετεύεται ξανά μέσα από τα δοχεία, θερμαίνεται και οδηγείται σε στροβίλους όπου εκτονώνεται παράγοντας έργο. Η διαδικασία της εκτόνωσης γίνεται χωρίς την χρήση καυσίμων.[21]



Εικόνα 9: Αδιαβατικό CAES

Προβλήματα υλοποίησης

Τα βασικότερα προβλήματα που προκύπτουν στην υλοποίηση του παραπάνω προγράμματος έχουν να κάνουν με: [21]

- Την έλλειψη τεχνογνωσίας σε συμπιεστές που να μπορούν να λειτουργήσουν σε υψηλές πιέσεις (100bar) με πολύ υψηλές θερμοκρασίες (600°C).
- Την έλλειψη τεχνογνωσίας σε στροβίλους που να μπορούν να λειτουργήσουν σε υψηλές πιέσεις (100bar) με πολύ υψηλές θερμοκρασίες (600°C).
- Την δυσκολία κατασκευής δοχείων ανάκτησης θερμότητας υψηλής πίεσης αλλά και την αντοχή τους σε καθημερινούς κύκλους

1.3 Διείσδυση ενέργειας στο δίκτυο

1.3.1 Ενσωμάτωση στο εθνικό δίκτυο

Το σημερινό δίκτυο παροχής ηλεκτρικού ρεύματος είναι ένα αρκετά σύνθετο σύστημα με πολλές παραμέτρους. Ο όρος "ποιότητα ενέργειας" χρησιμοποιείται για να περιγράψει τη συσχέτιση μεταξύ των παραδοσιακών μέσων παραγωγής

ενέργειας μέσω συμβατικών καύσιμων, πυρηνικών, ενέργειας από ανανεώσιμες πηγές και των καταναλωτών-αγοραστών (μεγάλων βιομηχανιών, οικιακών καταναλωτών κλπ). Τα τελευταία 20 και πλέον χρόνια όλο και περισσότερες μέθοδοι αξιοποίησης των ΑΠΕ αρχίζουν να αναπτύσσονται και ανάλογη είναι και η αύξηση των παραγωγών που προστίθενται στα δίκτυα διανομής. Ως γνωστόν, βασικός γνώμονας είναι η αντιστοιχία μεταξύ παραγωγής και ζήτησης, όταν όμως αυτό δεν είναι δυνατόν στρεφόμαστε στην αποθήκευση ηλεκτρικής ενέργειας μέσω μετατροπής της σε άλλο είδος όπως, θερμική, χημική, δυναμική κλπ. Σε τοπικό επίπεδο, η απόκλιση της τάσης είναι το βασικό πρόβλημα που σχετίζεται με την αιολική ενέργεια ενώ τα επίπεδα ανοχών είναι της τάξης του +/-10%.[11]

1.3.2 Διακυμάνσεις στην τάση και το "flickering"

Η διακύμανση στην τιμή της τάσης του ρεύματος οφείλεται στη μεταβολή του φορτίου ή/ και στο γεγονός ότι η παραγόμενη ενέργεια δεν είναι αρκετά "ποιοτική". Μεγάλες διακυμάνσεις στην τάση μπορούν να προκληθούν από ενεργοβόρες διεργασίες όπως χύτευση μετάλλων, ηλεκτροσυγκόλληση και εκκίνηση μεγάλων ηλεκτροκινητήρων. Μικρές εναλλαγές στην τάση εντός εύρους -10+6% εμφανίζονται συχνά αλλά δεν προκαλούν πρόβλημα.

Οι σύντομες και μικρές μεταβολές της τάσης ονομάζονται flickering. Οι μεταβολές αυτές αξιολογούνται με βάση τον κανονισμό IEC 1000-3-7, ο οποίος ορίζει τα όρια των μεταβαλλόμενων φορτίων στα δίκτυα μέσης τάσης (από 1 έως 36kV) και υψηλής τάσης (36-230kV). Η αξιολόγηση γίνεται με βάση μια καταγεγραμμένη καμπύλη η οποία απεικονίζει το όριο εμφάνισης ορθογώνιας μεταβολής της τάσης σε έναν λαμπτήρα πυρακτώσεως. Οριακά ορατές μεταβολές τάσης θεωρούνται ότι έχουν συντελεστή βαρύτητας Pst = 1. Ο αντίστοιχος συντελεστής στις μεγάλες μεταβολές της τάσης (Plt) υπολογίζεται από τον παρακάτω τύπο:

$$P_{lt} = \sqrt[3]{\frac{1}{12}\sum_{j=1}^{j-3} P_{stj-3}}$$

Ο δείκτης P_{st} μετριέται κάθε 10 λεπτά και ο δείκτης P_{lt} είναι έγκυρος για διάστημα 2 ωρών. [11]

1.3.3 Συχνότητα

Η παροχή και διανομή της ενέργειας βασίζεται παγκοσμίως σε εναλλασσόμενο ρεύμα (AC). Ο αριθμός των εναλλαγών μεταξύ της θετικής και αρνητικής πολικότητας και της διεύθυνσης του ρεύματος στη μονάδα του χρόνου ορίζεται ως

συχνότητα, μετράται σε Hertz (Hz) και για τα Ευρωπαϊκά δίκτυα είναι 50Hz ενώ της Αμερικής 60Hz. Η συχνότητα είναι ανάλογη της ταχύτητας περιστροφής των σύγχρονων γεννητριών, γεγονός το οποίο είναι βασικός παράγοντας σχεδιασμού της μηχανής.

Οι απαιτήσεις για τον έλεγχο της συχνότητας στην Δυτική Ευρώπη καταγράφονται στους κανόνες UCPTE (Union for the Co-ordination of Production and Transmission of Electricity). Η περιοχή χωρίζεται σε έναν αριθμό ζωνών ελέγχου, η κάθε μια από τις οποίες έχει ένα τμήμα πρωτεύοντος ελέγχου που είναι υπεύθυνο για την απόκλιση της συχνότητας με σκοπό να κρατά την ισορροπία μεταξύ της άμεσης και της συνολικής παραγόμενης ενέργειας. Το τμήμα δευτερεύοντος ελέγχου στοχεύει στην εξισορρόπηση της παραγωγής και της ζήτησης μέσα στις αυτόνομες ζώνες και μεταξύ όλων των ζωνών μεταξύ τους.

Το ποσό της ισχύος που απαιτείται για τον πρωτογενή έλεγχο ανέρχεται στα 3000MW και διαμερίζεται διαμέσου των ζωνών ελέγχου όπου η συχνότητα καταγράφεται με ρολόγια που παρακολουθούν τις ελάχιστες διακυμάνσεις της συχνότητας έτσι ώστε η μέση τιμή της να είναι τα 50Hz. [11]

1.3.4 Σταθερότητα του συστήματος

Όσον αφορά τη σταθερότητα του συστήματος τρεις είναι οι πλέον σημαντικοί παράμετροι που μας απασχολούν: Η υπερφόρτωση, η απώλεια ισχύος της παραγωγής και το βραχυκύκλωμα.

Η υπερφόρτωση του δικτύου μεταφοράς ή η βλάβη σε κάποιο τμήμα αυτής, διαταράσσει την ισορροπία του συστήματος και αυτή μπορεί να μεταφερθεί σε παρακείμενα δίκτυα. Ακόμα και αν υπάρχει επάρκεια παραγωγής από τις μονάδες η ανισορροπία μπορεί να προκαλέσει μεγάλη πτώση τάσης, η οποία δύναται να ωθήσει το δίκτυο εκτός των ορίων λειτουργίας τους προκαλώντας πλήρη πτώση ισχύος (blackout).

Μια βλάβη σε μονάδα παραγωγής θα προκαλέσει στιγμιαία ανισορροπία ισχύος στο δίκτυο. Αν οι υπόλοιπες μονάδες δεν έχουν την δυνατότητα να καλύψουν αυτή την απώλεια μέσα σε μικρό χρονικό διάστημα, θα εμφανιστεί μεγάλη πτώση τάσης και συχνότητας η οποία θα προκαλέσει ολική απώλεια ισχύος (blackout). Ένας τρόπος αποφυγής του blackout είναι να αποσυνδεθεί από το δίκτυο μια περιοχή ή κάποιοι μεγάλοι καταναλωτές ώστε να μειωθεί το φορτίο έως ότου αποκατασταθεί η ισορροπία του συστήματος μέσω βοηθητικών μονάδων.

Οι συχνότερες βλάβες σε ένα δίκτυο εμφανίζονται λόγω βραχυκυκλωμάτων στα δίκτυα μεταφοράς. Πολλές από αυτές τις βλάβες διορθώνονται άμεσα μέσω ασφαλιστικών διατάξεων (relay) ή με διακοπή και επανασύνδεση του δικτύου εντός

σύντομου χρονικού διαστήματος (milliseconds). Σε κάθε περίπτωση το αποτέλεσμα είναι η μερική ή ολική πτώση τάσης για μικρό χρονικό διάστημα. Εάν ένα τέτοιο γεγονός λάβει χώρα σε περιοχή όπου υπάρχει αιολικό πάρκο συνδεδεμένο στο δίκτυο, οι ελεγκτές του πάρκου θα εκλάβουν την πτώση τάσης ως βλάβη και θα αποσυνδέσουν το πάρκο από το δίκτυο για να το προστατέψουν από πιθανή βλάβη. Αυτό θα επιτείνει το πρόβλημα μιας και θα αφαιρέσει μια παραγωγική μονάδα από το δίκτυο. Η αντιμετώπιση του παραπάνω προβλήματος προβλέπει την τοποθέτηση ελεγκτών στα πάρκα, εξάρτημα το οποίο δεν τοποθετείται μέχρι σήμερα. [11]

1.4 Υβριδικά συστήματα αποθήκευση ενέργειας

Σκοπός των υβριδικών συστημάτων αποθήκευσης ενέργειας είναι η κάλυψη των ενεργειακών αναγκών μιας περιοχής ή αυτόνομων καταναλωτών. Ο συνδυασμός ανανεώσιμων πηγών ενέργειας με διάφορα συστήματα αποθήκευσης μπορούν να αποτελέσουν μια αποτελεσματική λύση.

1.4.1 Μικρά συστήματα

Στα μικρά συστήματα η ηλεκτροχημική μέθοδος αποθήκευσης σε συνδυασμό με τον κατάλληλο συνδυασμό αιολικής και ηλιακής ενέργειας στην ηλεκτροπαραγωγή μπορεί να αποτελέσει μια αξιόπιστη και οικονομική λύση. Οι συσσωρευτές μπορεί να είναι μολύβδου, στοιχείων ροής ή ιόντων λιθίου κλπ. Αυτά τα συστήματα αποθήκευσης μπορούν να αναπτυχθούν ως αυτόνομα ή ως ένα (ή και περισσότερα) κεντρικό σύστημα αποθήκευσης σε συνεργασία με αιολικές μονάδες. [1]

Εικόνα 10: Φωτοβολταϊκό πάρκο

Εικόνα 11: Αιολικό πάρκο

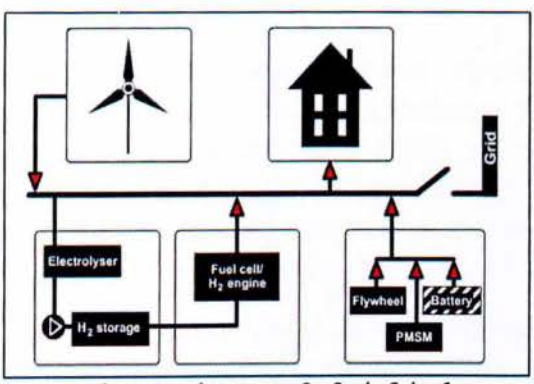
1.4.2 Μεσαία και μεγάλα συστήματα

Στα μεσαία και τα μεγάλα συστήματα εντάσσονται οι αναστρέψιμοι υδροηλεκτρικοί σταθμοί σε συνεργασία με αιολικά πάρκα όπως και τα CAES στα οποία αναφερθήκαμε παραπάνω. Τα πρώτα προσφέρονται για κεντρικά συστήματα αποθήκευσης αφού το σύνθετο ανάγλυφο με τους ορεινούς όγκους και το υψηλό αιολικό δυναμικό στις ορεινές περιοχές ευνοεί την ανάπτυξή τους. [7]

Εικόνα 12: Αναστρέψιμο υδροηλεκτρικό

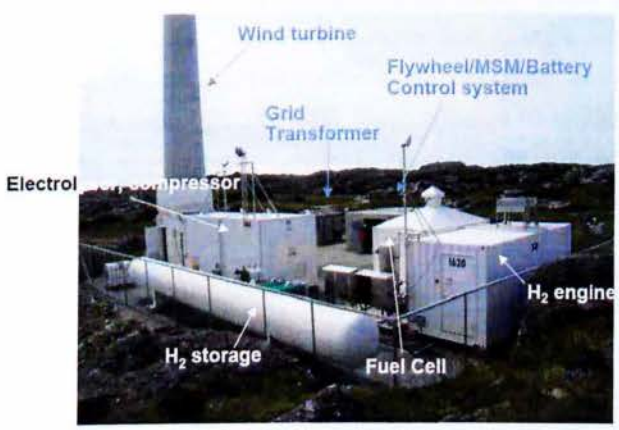
1.4.3 Υβριδικά συστήματα με κυψέλες καυσίμου

Μια άλλη μέθοδος που χρησιμοποιείται στην αποθήκευση της παραγόμενης ενέργειας από ανεμογεννήτριες είναι οι κυψέλες καυσίμου. Μέσω αυτής της μεθόδου εκμεταλλευόμαστε την ενέργεια του πάρκου για την ηλεκτρόλυση.



Εικόνα 13: Διάγραμμα υβριδικής διάταξης

Η πρώτη εγκατάσταση παραγωγής ενέργειας με αιολικό πάρκο και κυψέλες καυσίμου βρίσκεται στο νησί Utsira δυτικά της Νορβηγίας.[14]



Εικόνα 14: Η μονάδα Utsira

2. CAES

Το συνεχώς αυξανόμενο ενδιαφέρον για την προώθηση της αποκεντρωμένης παραγωγής ενέργειας με την χρήση τεχνολογίας ανανεώσιμων πηγών ενέργειας καλείται να διαδραματίσει έναν πολύ σημαντικό ρόλο στην επιχειρούμενη στροφή από τις κεντρικές μονάδες παραγωγής. Ταυτόχρονα, σε πολλά μέρη ανά την υφήλιο υπάρχουν περιοχές που δεν μπορούν να συνδεθούν στο κεντρικό δίκτυο διανομής ενέργειας και εξαρτώνται από ανεξάρτητες μονάδες παραγωγής που λειτουργούν με εισαγόμενο κυρίως πετρέλαιο.[6] Από την άλλη, σε αρκετές από αυτές τις περιοχές υπάρχουν καλές προϋποθέσεις για την ανάπτυξη παραγωγής ενέργειας από ανανεώσιμες πηγές όπως τα φωτοβολταϊκά και οι ανεμογεννήτριες. Παρότι οι εν λόγω τεχνολογίες θεωρούνται πλέον αρκετά ανεπτυγμένες, απαιτείται η χρήση εφεδρικών διατάξεων για την κάλυψη των μεταβαλλόμενων φορτίων και της στοχαστικής φύσης του ανέμου έτσι ώστε να επιτευχθεί η πλήρης ενεργειακή αυτονομία. [4]

Όπως αναφέρθηκε παραπάνω υπάρχουν αρκετές τεχνολογίες αποθήκευσης ενέργειας. Το ενδιαφέρον όμως το τελευταίο διάστημα έχει εστιαστεί στην αποθήκευση ενέργειας με συμπιεσμένο αέρα (CAES). Η λειτουργία αυτού του συστήματος βασίζεται στην λογική της χρήσης της περίσσειας ενέργειας (αιολικών πάρκων) ή την αγορά ενέργειας χαμηλού κόστους λόγω μη αιχμής και την χρήση της για συμπίεση αέρα υπό υψηλή πίεση σε υπόγειες κοιλότητες ή δοχεία πίεσης.

Κατά τις ώρες αιχμής ή σε περίπτωση που η παραγωγή από ανανεώσιμες πηγές δεν μπορεί να καλύψει την ζήτηση, ο αποθηκευμένος αέρας ανακτάται, αναμειγνύεται με φυσικό αέριο και καίγεται σε θάλαμο καύσης για την παραγωγή καυσαερίων υψηλής ενθαλπίας, τα οποία θα εκτονωθούν σε αεριοστρόβιλο με σκοπό να δώσουν μηχανικό έργο, το οποίο μέσω γεννήτριας μετατρέπεται σε ηλεκτρικό ρεύμα. Το όφελος από την παραπάνω διαδικασία είναι διπλό, μιας και εκμεταλλευόμαστε περίσσεια ενέργειας η οποία υπό άλλες συνθήκες θα χανόταν ανεκμετάλλευτη και μειώνουμε την κατανάλωση καυσίμου σε σύγκριση με έναν συμβατικό κύκλο αεριοστροβίλου (κύκλος Brayton) όπου το καθαρό έργο μειώνεται σημαντικά από το γεγονός ότι περίπου τα 2/3 της ισχύος του στροβίλου χρησιμοποιούνται για την κίνηση του συμπιεστή. Γνωρίζοντας τα οφέλη από την λειτουργία ενός τέτοιου συστήματος και δεδομένου ότι το CAES θεωρείται αυτόνομο σύστημα αποθήκευσης ενέργειας, είναι δυνατόν να το χρησιμοποιήσουμε σε μικρότερη κλίμακα για την κάλυψη των αναγκών σε μικρού και μεσαίου μεγέθους εφαρμογές. Η έρευνα αυτή εξετάζει την χρήση αιολικού πάρκου σε συνδυασμό με CAES για την κάλυψη των αναγκών απομακρυσμένων περιοχών.

Πιο συγκεκριμένα, μέσω αλγορίθμου βελτιστοποίησης που αναπτύχθηκε με βασική προϋπόθεση την ενεργειακή αυτονομία μελετάται η συμπεριφορά του CAES. Επιπροσθέτως, για την αξιολόγηση της επίδρασης του αιολικού δυναμικού στα αποτελέσματα, θα εξεταστούν διαφορετικά αιολικά δυναμικά τόσο ποιοτικά όσο και ποσοτικά. Επίσης, μελετώνται δύο διαφορετικές εκδόσεις, μια με συμβατικό CAES και μια με Dual-mode CAES, στο οποίο το σύστημα επιτρέπει την εναλλαγή σε απλό κύκλο Brayton μέσω συμπλέκτη όταν τα αποθέματα του συμπιεσμένου αέρα δεν επαρκούν για την κάλυψη της ζήτησης. Τα αποτελέσματα αναδεικνύουν την βιωσιμότητα της προτεινόμενης λύσης και την ανάγκη για υψηλό αιολικό δυναμικό ώστε να επιτευχθεί αυτονομία. Τέλος, φαίνεται καθαρά ότι σε περιοχές με χαμηλό αιολικό δυναμικό θα πρέπει να προτιμηθεί η λύση του Dual-mode CAES.

*2.1 Περιγραφή λειτουργίας του **CAES***

Κατά την λειτουργία του CAES η περίσσεια ενέργειας του δικτύου χρησιμοποιείται για την συμπίεση αέρα σε υψηλή πίεση (70-100bar) και την αποθήκευση του σε υπόγειες κοιλότητες (αεροθυλάκια) ή δοχεία πίεσης. Σε περιόδους που εμφανίζεται αυξημένη ζήτηση για ενέργεια, ο συμπιεσμένος αέρας διοχετεύεται σε θάλαμο καύσης όπου αναμειγνύεται με φυσικό αέριο, καίγεται και διοχετεύεται σε αεριοστρόβιλο όπου εκτονώνεται. Το βασικό πλεονέκτημα του συστήματος είναι η ανεξάρτητη λειτουργία της φάσης συμπίεσης και της φάσης παραγωγής έργου. Έτσι, επιτυγχάνεται οικονομία καυσίμου της τάξης του 66% μιας και δεν δαπανάται έργο από τον στρόβιλο για την συμπίεση. Έτσι, όλο το έργο που παράγεται στον στρόβιλο είναι διαθέσιμο για την παραγωγή ενέργειας προς το δίκτυο. Παρότι η μείωση της κατανάλωσης είναι αξιοσημείωτη, το σύστημα δεν παύει να εξαρτάται από ορυκτά καύσιμα. Για την απαγκίστρωση από αυτά είναι δυνατή η χρήση βιοκαυσίμων. Επίσης, γίνεται προσπάθεια για την δημιουργία αδιαβατικού CAES, η λειτουργία του οποίου περιγράφεται αναλυτικά παραπάνω. Η χωροθέτησή μιας μονάδας CAES απαιτεί την ύπαρξη υπόγειων κοιλοτήτων που να μπορούν να χρησιμοποιηθούν ως αποθηκευτικοί χώροι για τον συμπιεσμένο αέρα. Συνήθως, χρησιμοποιούνται πετρώδεις κοιλότητες ή παλιά ορυχεία αλάτων, όμως για μικρές μονάδες μπορούν να χρησιμοποιηθούν και δοχεία πίεσης ή θαμμένοι αγωγοί.

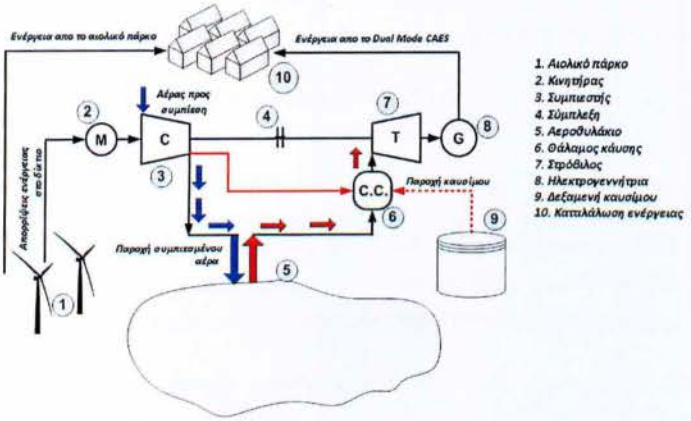
Τα CAES θεωρούνται αξιόπιστες εναλλακτικές λύσεις των αναστρέψιμων υδροηλεκτρικών. Μιας και οι απώλειες κατά την αποθήκευση θεωρούνται αμελητέες, η χρονική διάρκεια την αποθήκευσης δεν έχει ουσιαστικό περιορισμό. Στα βασικότερα πλεονεκτήματα των CAES περιλαμβάνονται ο μικρός χρόνος αντίδρασης του συστήματος (2-3 φορές πιο σύντομος από τις συμβατικές μονάδες), η σταθερή ειδική κατανάλωση καυσίμου σε χαμηλό φορτίο και οι μειωμένες εκπομπές αέριων ρύπων. Επίσης, λόγω της δυνατότητας του να λειτουργεί με μερικό φορτίο πετυχαίνοντας χαμηλές καταναλώσεις καυσίμου, τα CAES μπορούν

να χρησιμοποιηθούν ως ρυθμιστές του δικτύου. Εκτός αυτού, τα CAES μπορούν να λειτουργήσουν είτε ως μονάδες βάσης, είτε ως μονάδες κάλυψης φορτίου αιχμής παρέχοντας την δυνατότητα καλύτερης διαχείρισης αιολικών πάρκων. [13]

2.2 Περιγραφή λειτουργίας του Dual-mode CAES

Εστιάζοντας στην αποκεντρωμένη παραγωγή ενέργειας αλλά και στην ανάγκη για ενεργειακή αυτονομία με την χρήση του συνδυασμού αιολικού πάρκου - CAES προτείνεται μια επιπλέον εναλλακτική λύση. [9] Για την αποφυγή υπερδιαστασιολόγησης του αιολικού πάρκου, ώστε να επιτευχθεί 100% αυτονομία στις απομονωμένες περιοχές, υιοθετείται η λύση του Dual-mode CAES. Η μονάδα αυτή έχει την δυνατότητα να μεταβάλλει την λειτουργία του από CAES σε κλασσικό κύκλο αεριοστροβίλου με την χρήση συμπλέκτη ο οποίος επιτρέπει την σύνδεση του συμπιεστή με τον στρόβιλο. Η λύση αυτή (Εικόνα 13) αποτελείται από τα ακόλουθα μέρη:

- Αιολικό πάρκο αποτελούμενο από ανεμογεννήτριες με συνολική ισχύ "N_{WP}".
- Ηλεκτροκινητήρα ισχύος "N_m" ο οποίος δίνει κίνηση στον συμπιεστή χρησιμοποιώντας την περίσσεια ενέργεια του αιολικού πάρκου με βαθμό απόδοσης "η_m".
- Πολυβάθμιος συμπιεστής με ισχύ "N_c" ο οποίος συμπιέζει ατμοσφαιρικό αέρα με σχέση συμπίεσης "r_c" και βαθμό απόδοσης "η_c".
- Αεροθυλάκιο ή δοχείο πίεσης με όγκο "V_{ss}" και μέγιστο βάθος εκφόρτισης "DOD_L"
- Θάλαμος καύσης όπου γίνεται η ανάμειξη και καύση του συμπιεσμένου αέρα με τον φυσικό αέριο σε θερμοκρασία "T_{cc}".
- Δεξαμενή αποθήκευσης φυσικού αερίου με θερμογόνο δύναμη "H_u".
- Αεριοστρόβιλο ισχύος "N_T", με βαθμό απόδοσης "η_T"



Εικόνα 15: Διάγραμμα λειτουργίας μονάδας

Ισχύς ηλεκτροκινητήρα "N_m"	$N_c(N_{WP}-N_{min})$
Απόδοση ηλεκτροκινητήρα "η_m"	0.90
Ισχύς συμπιεστή "N_c" (MW)	30
Απόδοση συμπιεστή "η_c"	0.70
Όγκος αεροθυλακίου "V_{ss}" (m^3)	Variable
Ισχύς αεριοστροβίλου "N_t" (MW)	Variable
Β.Α αεριοστροβίλου "η_T"	0.80
Β.Α στροβίλου Brayton "η_{GT}"	0.35
Αναλογία αέρα- καυσίμου "λ"	3
Θερμογόνος δύναμη "H_u" (MJ/kg)	50
Θερ/σία περιβάλλοντος "T_{amb}" (K)	Time series
Θερ/σία αεροθυλακίου. "T_{ss}" (K)	300
Πίεση περιβάλλοντος "P_{amb}" (bar)	1
Σχέση συμπίεσης "r_c"	70
Σχέση εκτόνωσης "r_t"	30
Θερμοχωρητικότητα αέρα "C_p" (kJ/kg.K)	1.004
Θερμοχωρητικότητα καυσαερίων "C_g" (kJ/kg.K)	1.1
Θερ/σία θαλάμου καύσης. "T_{cc}" (K)	1300

Πίνακας 4: Μεταβλητές λειτουργίας μονάδας

Τα σενάρια λειτουργίας της παραπάνω μονάδας είναι τα εξής: [27]

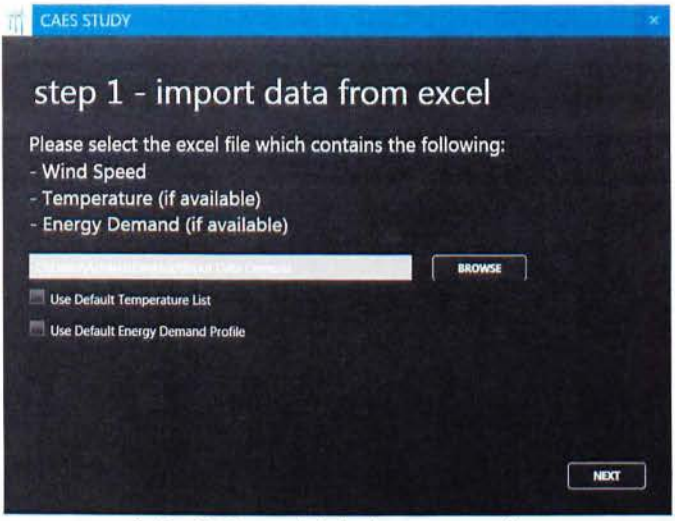
A. Όταν η παραγόμενη ενέργεια από το αιολικό πάρκο αρκεί για την κάλυψη της ζήτησης, αυτή οδηγείται απ' ευθείας προς την κατανάλωση. Αν υπάρχει περίσσεια ενέργειας, αυτή χρησιμοποιείται από τον συμπιεστή για αποθήκευση αέρα εφόσον το αεροθυλάκιο δεν είναι γεμάτο.
B. Όταν η παραγόμενη ενέργεια από το αιολικό πάρκο δεν επαρκεί για την κάλυψη της ζήτησης, το έλλειμμα καλύπτεται από την εκμετάλλευση του συμπιεσμένου αέρα και του καυσίμου.
C. Σε περίπτωση που κανένα από τα παραπάνω σενάρια δεν μπορούν να εφαρμοστούν, τότε το έλλειμμα καλύπτεται μέσω του απλού κύκλου Brayton, δηλαδή με απευθείας σύμπλεξη του συμπιεστή και του στροβίλου.

Τα παραπάνω σενάρια ελέγχονται σε ωριαία βάση.

3. Το πρόγραμμα

Για την αναλυτική και σε βάθος μελέτη του CAES ώστε να βρεθεί η λύση που να προσφέρει πλήρη ενεργειακή αυτονομία μελετώντας όλες τις μεταβλητές που το επηρεάζουν αναπτύχθηκε αλγόριθμος στην γλώσσα προγραμματισμού C#.

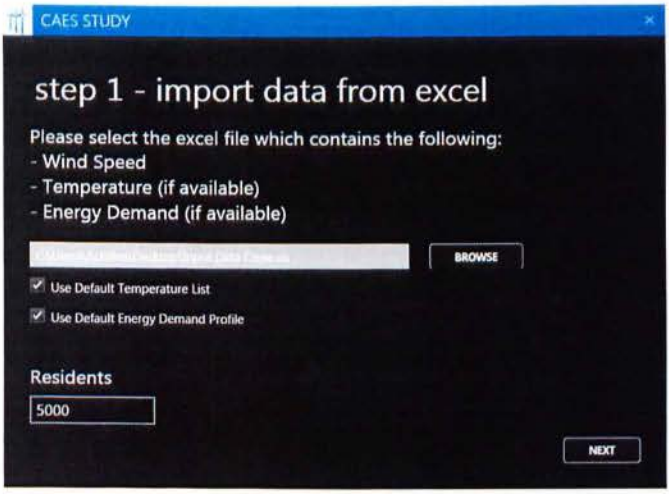
Το πρόγραμμα απαιτεί ως δεδομένα μια χρονοσειρά με ωριαίες μετρήσεις ταχύτητας ανέμου (m/s), θερμοκρασίας (°C) και ζήτησης ενέργειας (MW) σε αρχείο excel τύπου xls (Εικόνα 16).



Εικόνα 16: Εισαγωγή δεδομένων χρονοσειράς

Σε περίπτωση που δεν υπάρχουν δεδομένα θερμοκρασιών και δεν απαιτείται απόλυτη ακρίβεια στα αποτελέσματα ο χρήστης μπορεί να επιλέξει την χρήση προεπιλεγμένης χρονοσειράς θερμοκρασιών που απαρτίζεται από μέσες ωριαίες θερμοκρασίες περιοχής με ήπιο κλίμα.

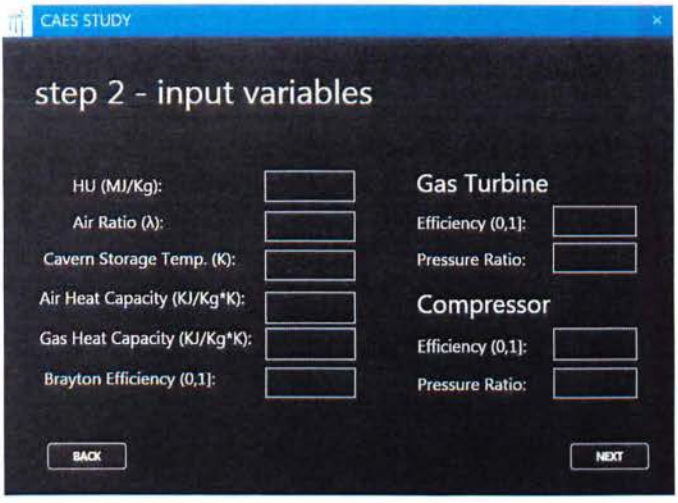
Επίσης, σε περίπτωση που δεν υπάρχουν δεδομένα ζήτησης ενέργειας αλλά είναι γνωστός ο πληθυσμός της περιοχής που μελετάται, υπάρχει επιλογή όπου ο χρήστης εισάγει τον αριθμό των κατοίκων και μέσω προεπιλεγμένης χρονοσειράς κατανάλωσης, δημιουργεί το δικό του προφίλ κατανάλωσης (Εικόνα 17).



Εικόνα 17: Χρήση προεπιλεγμένων χρονοσειρών

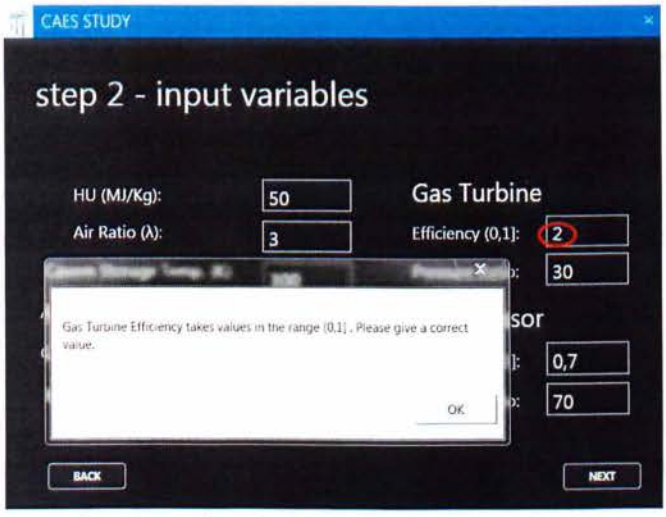
Σημείωση: Η μετάβαση από το Βήμα 1 στο βήμα 2 απαιτεί χρόνο ώστε να φορτωθούν οι χρονοσειρές στο πρόγραμμα.

Στην συνέχεια ο χρήστης εισάγει τις υπόλοιπες παραμέτρους που αφορούν την λειτουργία της μονάδας (Εικόνα 18).



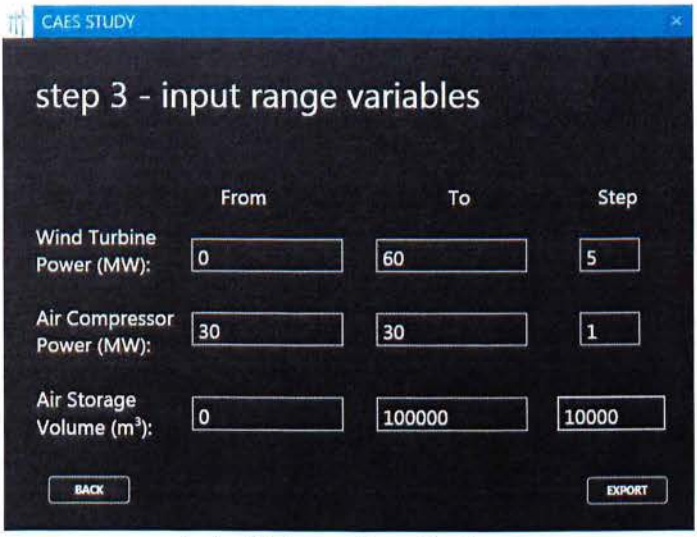
Εικόνα 18: Εισαγωγή παραμέτρων λειτουργίας

Σε περίπτωση εισαγωγής λανθασμένων μεγεθών όπως πχ θερμοκρασία -100K ή βαθμό απόδοσης εκτός του εύρους (0,1] το πρόγραμμα επιστρέφει μήνυμα λάθους και απαιτεί την διόρθωση του ώστε να προχωρήσει στο επόμενο βήμα. (Εικόνα 19)



Εικόνα 19: Αναγνώριση λανθασμένης εισαγωγής παραμέτρου

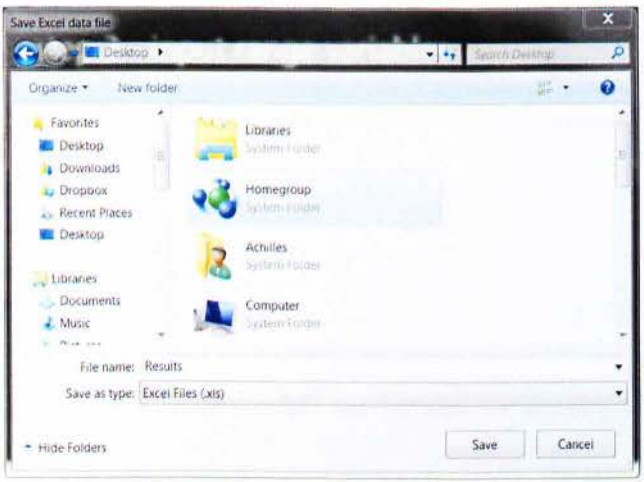
Εφόσον όλες οι παράμετροι έχουν εισαχθεί σωστά, περνάμε στο τελευταίο βήμα όπου απαιτείται η εισαγωγή των μεταβλητών. Υπάρχει η δυνατότητα μελέτης εύρους των παρακάτω μεγεθών ή και μεμονωμένων τιμών (Εικόνα 20).



Εικόνα 20: Εισαγωγή μεταβλητών

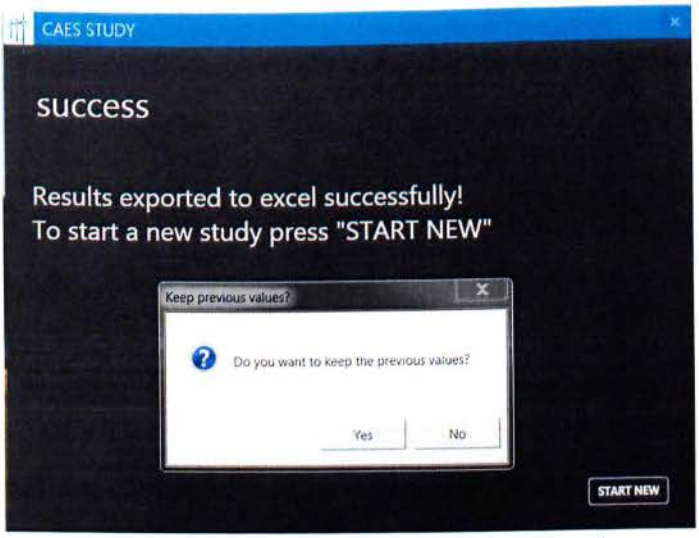
Στην συνέχεια το πρόγραμμα υπολογίζει τα αποτελέσματα με βάση τα δεδομένα και τα εξάγει στην τοποθεσία που υποδεικνύει ο χρήστης (Εικόνα 21).

Σημείωση: Η μετάβαση από το Βήμα 3 στη εξαγωγή αποτελεσμάτων απαιτεί χρόνο ώστε να υπολογιστούν όλα τα πιθανά σενάρια λύσεων.



Εικόνα 21: Αποθήκευση αποτελεσμάτων

Μετά την αποθήκευση των αποτελεσμάτων υπάρχει η δυνατότητα νέας μελέτης διατηρώντας τις ίδιες παραμέτρους (Εικόνα 22).



Εικόνα 22: Νέα μελέτη με διατήρηση των προηγούμενων παραμέτρων

Τα αποτελέσματα που εξήχθησαν από το πρόγραμμα αποθηκεύονται σε αρχείο excel της μορφής .xls και περιέχουν τόσο τις δοθείσες παραμέτρους όσο και τα αποτελέσματα.

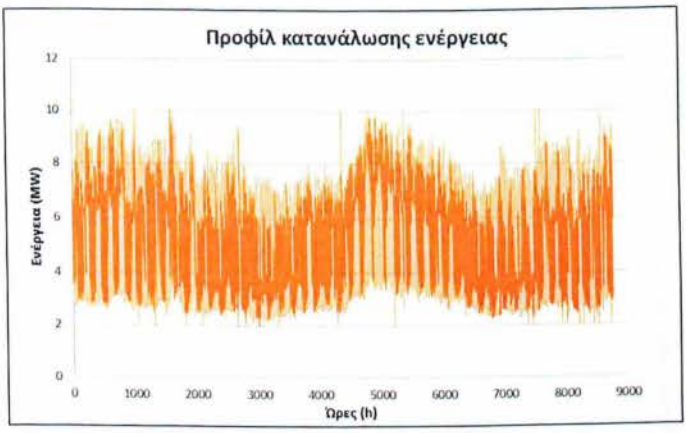
4. Οι μελετώμενες περιοχές

Η μελέτη της ενεργειακής αυτονομίας με την χρήση CAES και αιολικού πάρκου έγινε για 5 νησιωτικές περιοχές στην Ελλάδα με διαφορετικό αιολικό δυναμικό η κάθε μια. Οι προς μελέτη περιοχές είναι οι ακόλουθες:

- Άνδρος (Υψηλό αιολικό δυναμικό)
- Κρήτη (Υψηλό αιολικό δυναμικό)
- Νάξος (Μέτριο αιολικό δυναμικό)
- Κέα (Χαμηλό αιολικό δυναμικό)
- Λήμνος (Χαμηλό αιολικό δυναμικό)

Θεωρήθηκε ότι η ζήτηση ενέργειας είναι ίδια για όλες τις περιοχές έτσι ώστε τα αποτελέσματα να είναι συγκρίσιμα μεταξύ τους.

Το προφίλ κατανάλωσης που χρησιμοποιήθηκε αντιστοιχεί σε περίπου 10.000 κατοίκους και δίνεται στο παρακάτω διάγραμμα:



Διάγραμμα 6: Ενεργειακό προφίλ ζήτησης

Το αιολικό δυναμικό κάθε περιοχής παρουσιάζει διαφορετικά ποιοτικά και ποσοτικά χαρακτηριστικά που παίζουν σημαντικό ρόλο στην μελέτη μας. Πιο αναλυτικά στο παραπάνω πίνακα παραθέτονται βασικά στατιστικά στοιχεία των περιοχών.

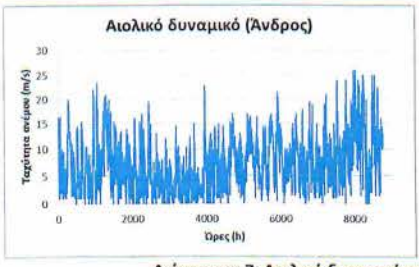
	Άνδρος	Κρήτη	Κέα	Λήμνος	Νάξος
Μέση τιμή ανέμου (m/s)	7,4	8,1	5,5	4,7	7,0
Τυπική απόκλιση	4,9	5,1	4,1	3,8	4,8
Αθροιστικά διαστήματα νηνεμίας (h)	2453	1909	3788	4326	2731
Διαστήματα συνεχούς νηνεμίας (h)	91	26	174	108	56

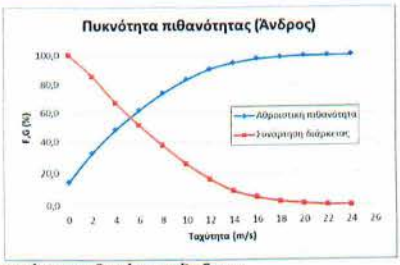
Πίνακας 5: Στατιστικά στοιχεία αιολικού δυναμικού περιοχών

Από τον παραπάνω πίνακα αξίζουν αναφοράς τα διαστήματα συνεχόμενης νηνεμίας κάθε περιοχής. Οι περιοχές με παρατεταμένα διαστήματα νηνεμίας έχουν την μεγάλη ανάγκη για αποθήκευση ενέργειας ώστε να επιτευχθεί αυτονομία. Παρατηρούμε ότι στην Κέα το διάστημα συνεχούς νηνεμίας διαρκεί πάνω από μια εβδομάδα, ενώ το αντίστοιχο διάστημα στην Κρήτη διαρκεί μια ημέρα. Άξιο αναφοράς είναι επίσης το αθροιστικό διάστημα νηνεμίας στην Λήμνο και στην Κέα, όπου για 180 και 158 ημέρες αντίστοιχα δεν είναι δυνατή η παραγωγή ενέργειας μέσω των ανεμογεννητριών.

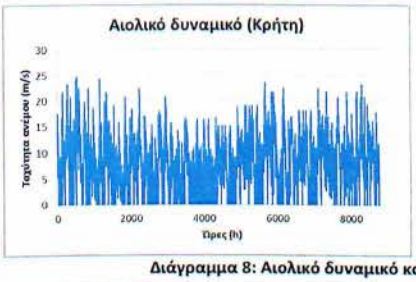
Για την καλύτερη σύγκριση των αιολικών δυναμικών υπολογίστηκε και η πυκνότητα πιθανότητας ανέμου για κάθε περιοχή.

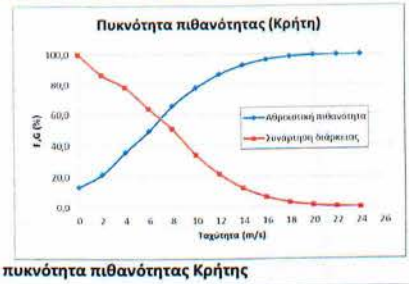
Από τα διαγράμματα πυκνότητας πιθανότητας αλλά και από την χρονοσειρά του αιολικού δυναμικού μπορούμε να δούμε τις ιδιαιτερότητες κάθε περιοχής και την δυνατότητα της να καλύψει τις ενεργειακές της ανάγκες από το αιολικό πάρκο.



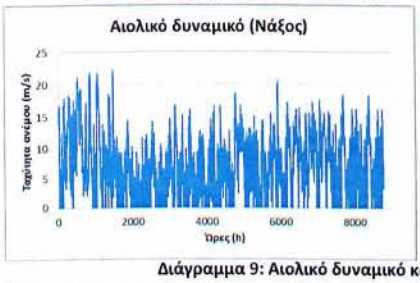


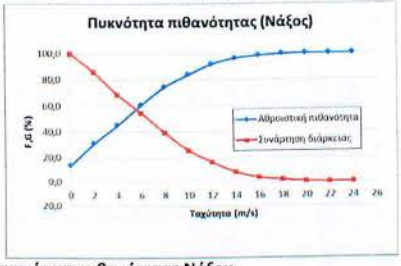
Διάγραμμα 7: Αιολικό δυναμικό και πυκνότητα πιθανότητας Άνδρου



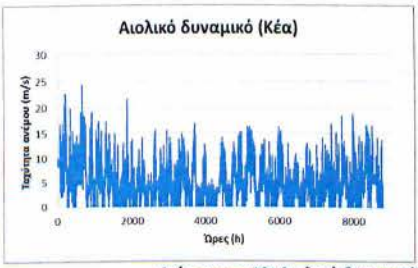


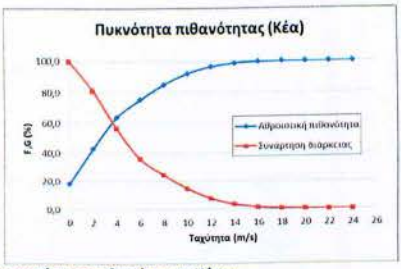
Διάγραμμα 8: Αιολικό δυναμικό και πυκνότητα πιθανότητας Κρήτης



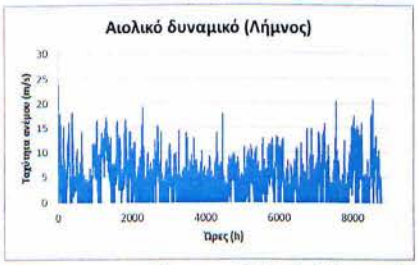


Διάγραμμα 9: Αιολικό δυναμικό και πυκνότητα πιθανότητας Νάξου



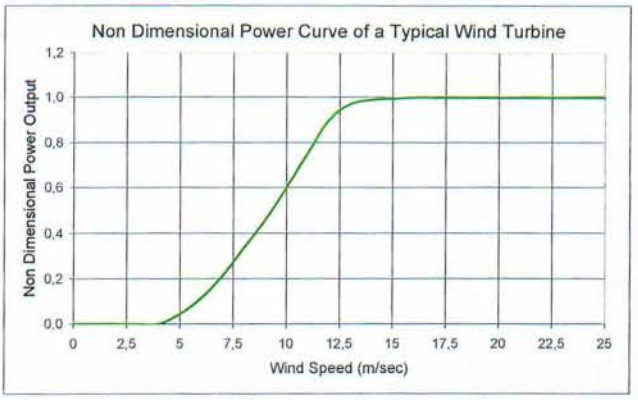


Διάγραμμα 10: Αιολικό δυναμικό και πυκνότητα πιθανότητας Κέας



Διάγραμμα 11: Αιολικό δυναμικό και πυκνότητα πιθανότητας Λήμνου

Παρακάτω παρατίθεται η αδιάστατη καμπύλη ισχύος της ανεμογεννήτριας που χρησιμοποιήθηκε στο αιολικό πάρκο. Η γεννήτρια που επιλέχθηκε είναι της εταιρείας Vestas, μοντέλο V80 με ονομαστική ισχύ 2MW.



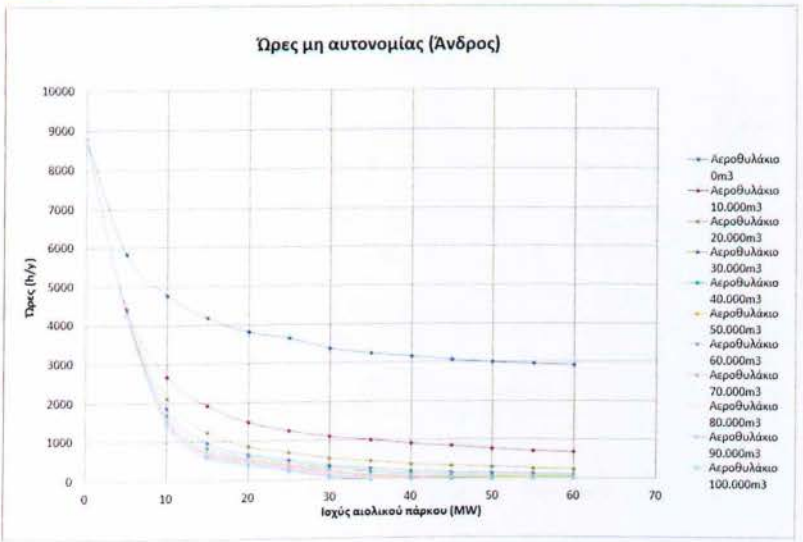
Διάγραμμα 12: Αδιάστατη ισχύς ανεμογεννήτριας

Με βάση τα παραπάνω δεδομένα και χρησιμοποιώντας τον αλγόριθμο που αναπτύχθηκε για το CAES εξήχθησαν τα παρακάτω αποτελέσματα σε μορφή διαγραμμάτων.

Οι υπολογισμοί έγιναν διατηρώντας σταθερή την ονομαστική ισχύ του συμπιεστή.

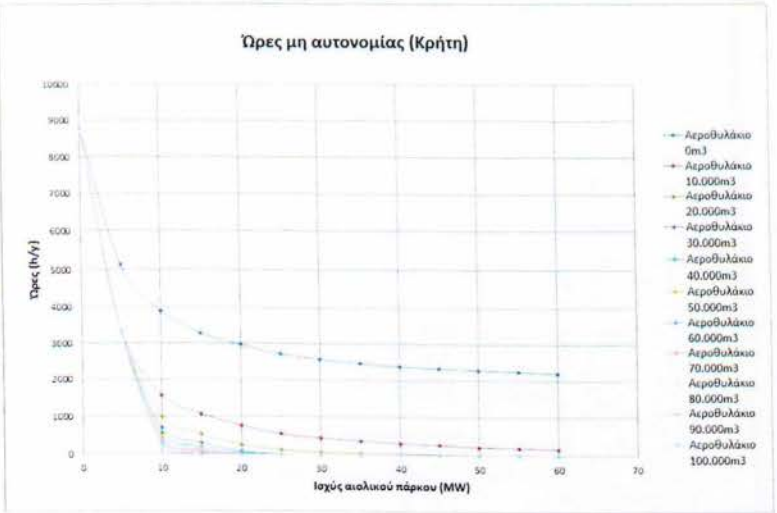
4.1 Ενεργειακή αυτονομία

Στην πρώτη ομάδα διαγραμμάτων εμφανίζεται η αυτονομία της κάθε περιοχής σε συνάρτηση με την ισχύ του αιολικού πάρκου και τον αποθηκευτικό χώρο του συμπιεσμένου αέρα.



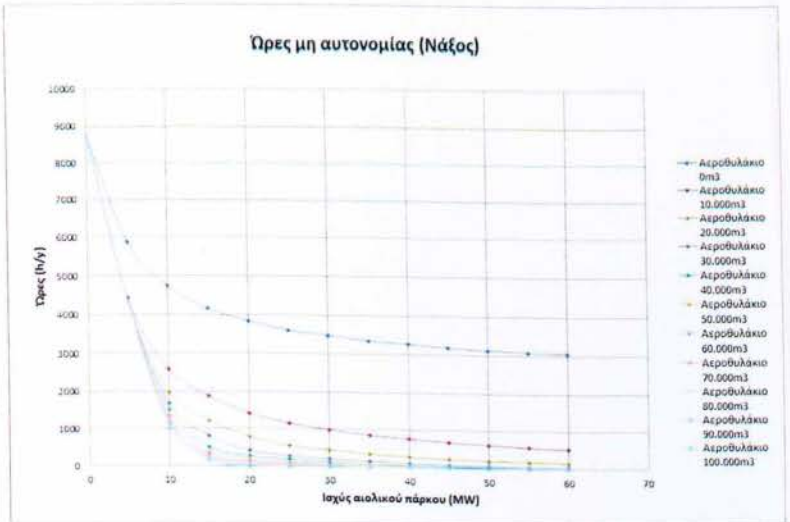
Διάγραμμα 13: Αυτονομία δικτύου στην Άνδρο

Από το παραπάνω διάγραμμα διαπιστώνεται ότι με αιολικό πάρκο 45MW και αεροθυλάκιο χωρητικότητας 70.000m^3 και άνω επιτυγχάνεται πλήρης ενεργειακή αυτονομία της περιοχής.



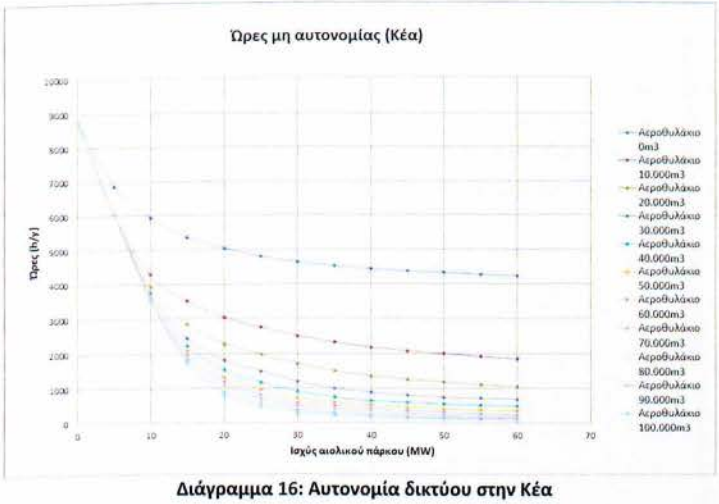
Διάγραμμα 14: Αυτονομία δικτύου στην Κρήτη

Αντιστοίχως για την περιοχή της Κρήτης παρατηρούμε ότι η αυτονομία επιτυγχάνεται με αιολικό πάρκο 30MW και όγκο αεροθυλακίου ίσο ή μεγαλύτερο από 30.000m^3.



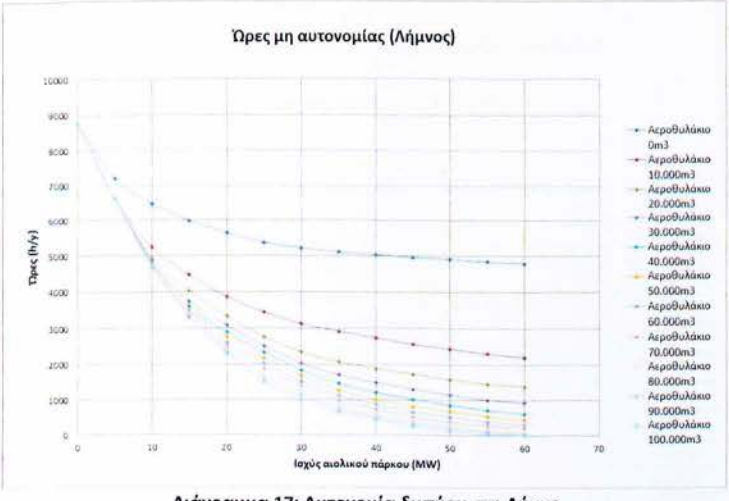
Διάγραμμα 15: Αυτονομία δικτύου στην Νάξο

Στην περιοχή της Νάξου η ενεργειακή αυτονομία επιτυγχάνεται με την χρήση αιολικού πάρκου 45MW και αεροθυλακίου χωρητικότητας 50.000m^3.



Διάγραμμα 16: Αυτονομία δικτύου στην Κέα

Το χαμηλό αιολικό δυναμικό σε συνάρτηση με μεγάλα διαστήματα συνεχούς νηνεμίας δεν επιτρέπουν να επιτευχθεί πλήρης ενεργειακή αυτονόμηση του νησιού. Ακόμα και στην ακραία περίπτωση της χρήσης 60MW αιολικού πάρκου και 100.000m^3 αεροθυλακίου, η αυτονομία δεν επιτυγχάνεται για 100 ώρες μέσα στο έτος. Για την αυτονόμηση του νησιού επιβάλλεται η χρήση Dual mode CAES.



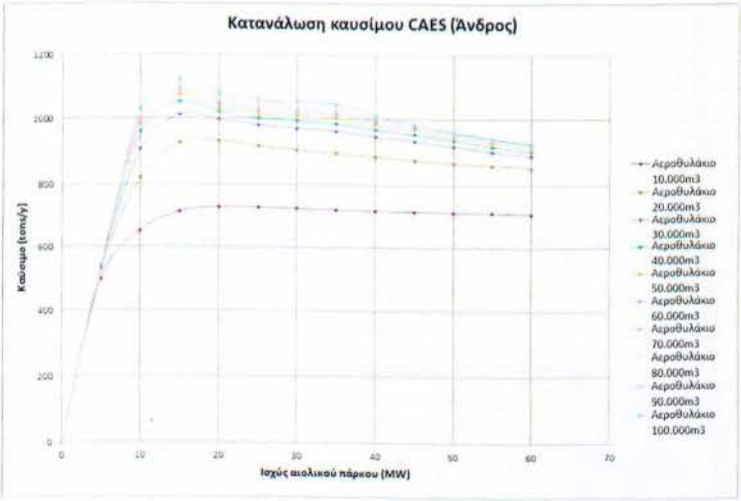
Διάγραμμα 17: Αυτονομία δικτύου στη Λήμνο

Στην περίπτωση της Λήμνου επιτυγχάνεται ενεργειακή αυτονομία μόνο με την χρήση 60MW αιολικού πάρκου και 100.000m^3 αεροθυλακίου, μεγέθη τα οποία

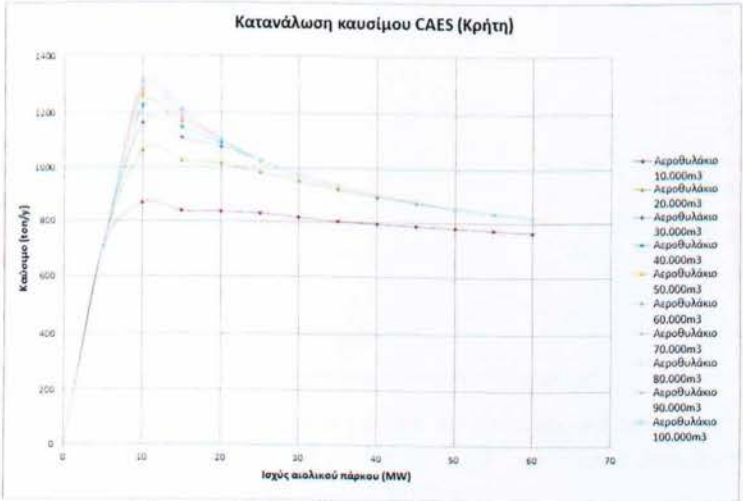
κρίνονται υπερβολικά για συγκεκριμένη ζήτηση ενέργειας. Και σε αυτή την περίπτωση κρίνεται απαραίτητη η χρήση Dual mode CAES.

4.2 Κατανάλωση ενέργειας CAES

Σε αυτή την ομάδα διαγραμμάτων παρατίθεται η κατανάλωση καυσίμου του CAES. Από το σύνολο των διαγραμμάτων μπορούμε να συμπεράνουμε ότι η κατανάλωση αυξάνεται όσο αυξάνεται και η ισχύς του πάρκου. Από ένα σημείο και μετά η κατανάλωση μειώνεται λόγω της συνεισφοράς του αιολικού πάρκου στην κάλυψη της ζήτησης.

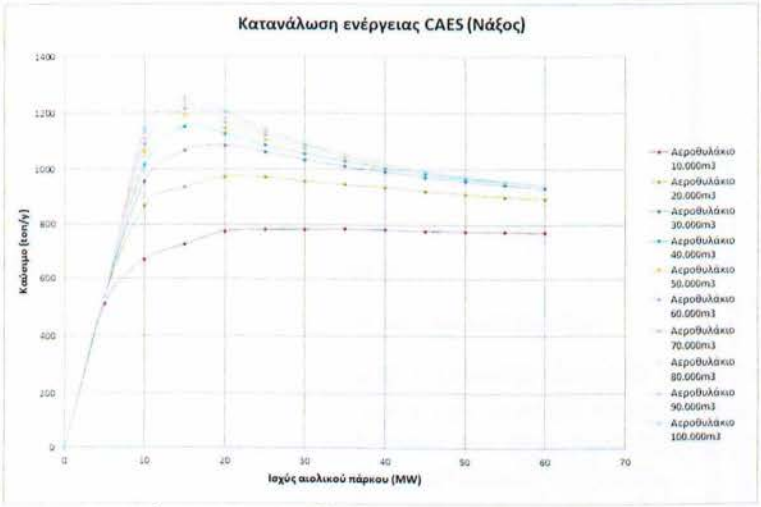


Διάγραμμα 18: Κατανάλωση καυσίμου CAES στην Άνδρο

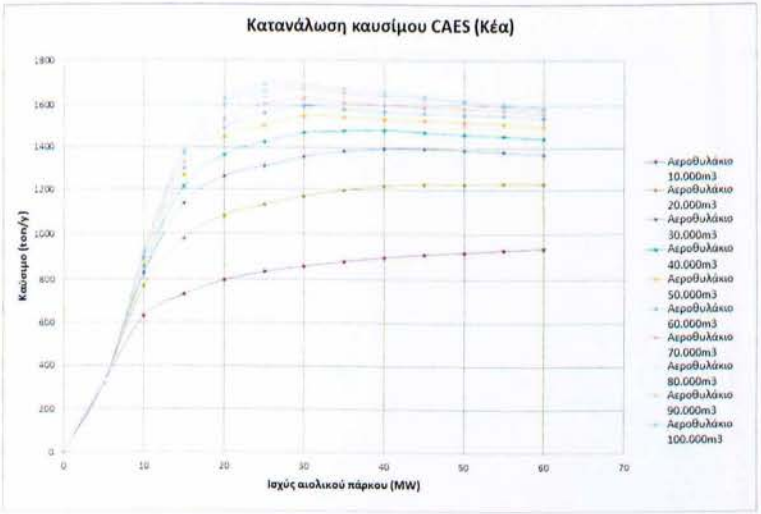


Διάγραμμα 19: Κατανάλωση καυσίμου CAES στην Κρήτη

Στις περιοχές όπου επιτυγχάνεται ενεργειακή αυτονομία (Άνδρος, Κρήτη, Νάξος) με την χρήση του CAES η μορφή του διαγράμματος είναι αυξητική αρχικά και στην συνέχεια μειώνεται αισθητά. Η μείωση οφείλεται στην αυξημένη συνεισφορά του αιολικού πάρκου στην κάλυψη της ζήτησης. Στις υπόλοιπες περιοχές (Λήμνος, Κέα) η καμπύλη είναι πολύ πιο ομαλή.

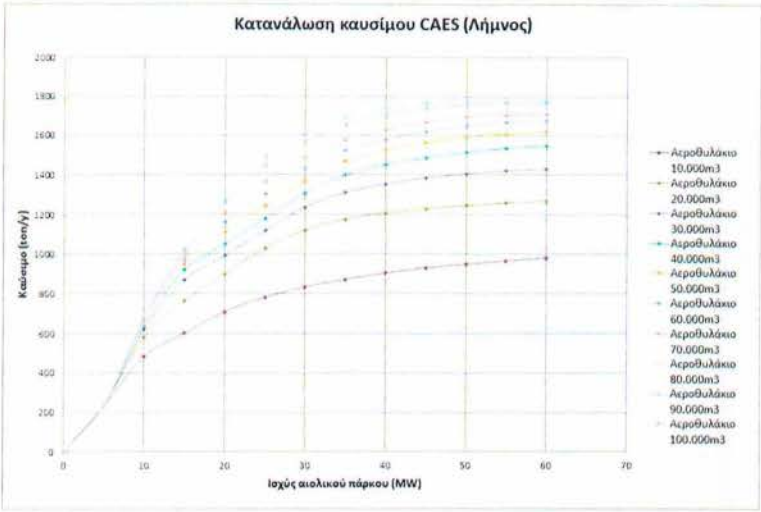


Διάγραμμα 20: Κατανάλωση καυσίμου CAES στη Νάξο



Διάγραμμα 21: Κατανάλωση καυσίμου CAES στην Κέα

Στο διάγραμμα της Κέας παρατηρούμε μείωση της κατανάλωσης όσο αυξάνεται η ισχύς του αιολικού πάρκου από τα 25MW και μετά. Αυτή όμως είναι μικρή σε σύγκριση με τις περιοχές με υψηλό αιολικό δυναμικό.

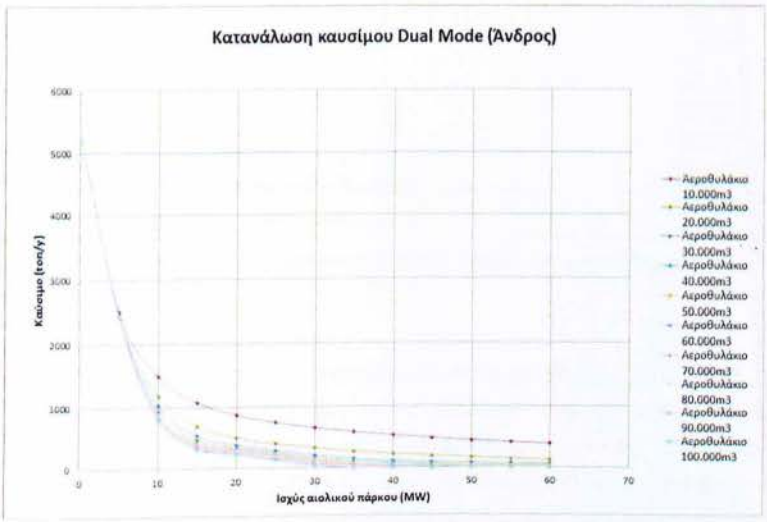


Διάγραμμα 22: Κατανάλωση καυσίμου CAES στη Λήμνο

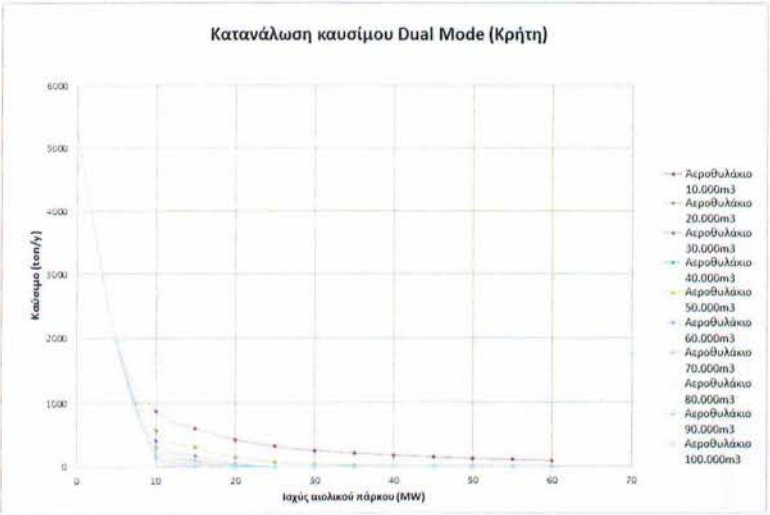
Στις περιοχές με χαμηλό αιολικό δυναμικό σημειώνεται αυξημένη κατανάλωση καυσίμου (1700-1800 τόνοι/έτος), ενώ στις περιοχές με υψηλό αιολικό δυναμικό η κατανάλωση κυμαίνεται περί τους 1000 τόνους ανά έτος. Αυτό οφείλεται στην αυξημένη συνεισφορά της αιολικής ενέργειας στην κάλυψη της ζήτησης.

4.3 Κατανάλωση ενέργειας Dual-mode CAES

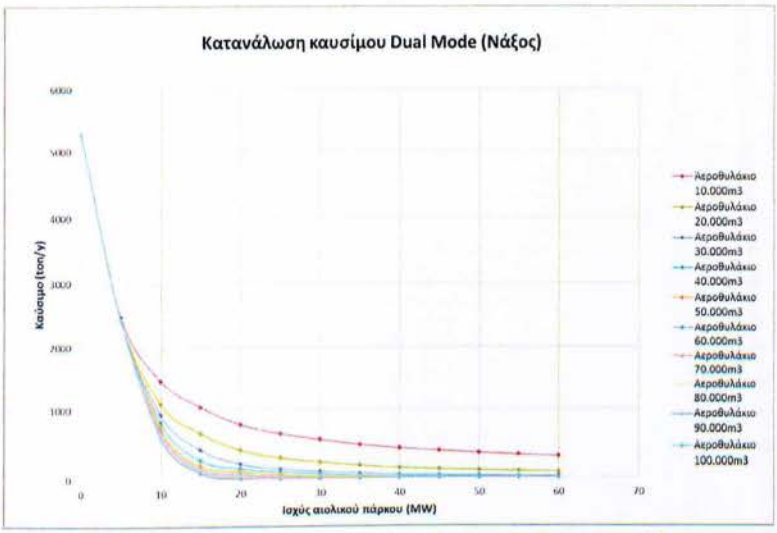
Σε αυτή την ομάδα διαγραμμάτων δίνεται η κατανάλωση καυσίμου μόνο κατά την λειτουργία του Dual-mode CAES.



Διάγραμμα 23: Κατανάλωση καυσίμου Dual-mode CAES στην Άνδρο

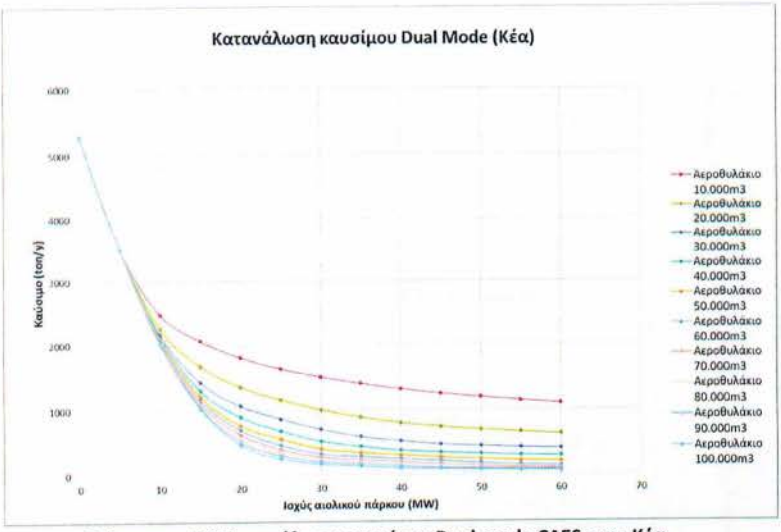


Διάγραμμα 24: Κατανάλωση καυσίμου Dual-mode CAES στην Κρήτη

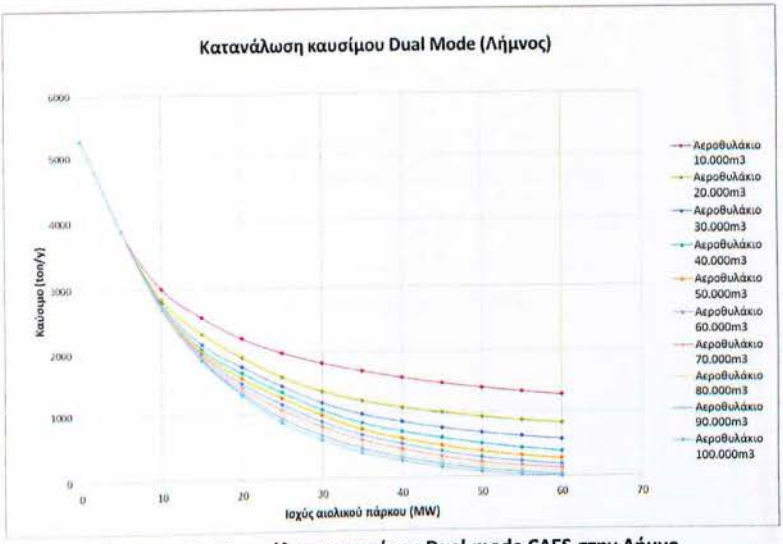


Διάγραμμα 25: Κατανάλωση καυσίμου Dual-mode CAES στην Νάξο

Από τα παραπάνω διαγράμματα συμπεραίνουμε ότι η χρήση του Dual-mode CAES δεν είναι απαραίτητη από μια τάξη ισχύος αιολικού πάρκου και άνω. Αυτό συμβαίνει διότι το αιολικό δυναμικό σε συνδυασμό με το CAES καλύπτουν το σύνολο της ζήτησης ενέργειας. Η χρήση του Dual-mode CAES είναι απαραίτητη μόνο όταν έχουμε μικρό όγκο αεροθυλακίου (<30.000m^3) ή μικρή ισχύ αιολικού πάρκου (<20-30MW).



Διάγραμμα 26: Κατανάλωση καυσίμου Dual-mode CAES στην Κέα

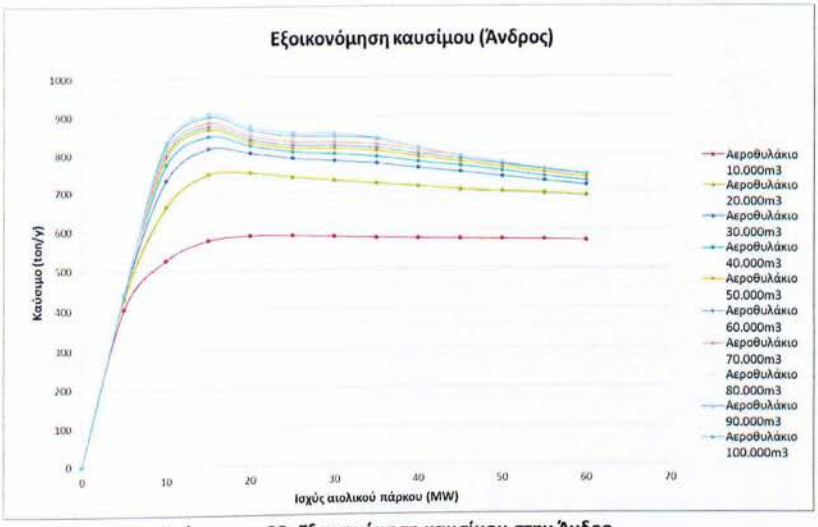


Διάγραμμα 27: Κατανάλωση καυσίμου Dual-mode CAES στην Λήμνο

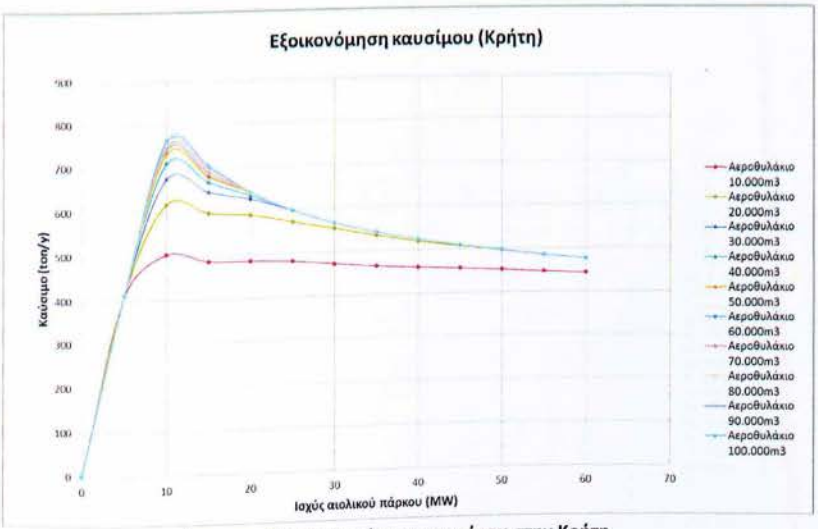
Στα δύο αυτά διαγράμματα (26 και 27) φαίνεται η ανάγκη για χρήση του Dual mode CAES για την κάλυψη της ζήτησης ενέργειας. Αυτό είναι αναμενόμενο μιας και το αιολικό δυναμικό της περιοχής δεν μπορεί να καλύψει την ζήτηση. Στην Λήμνο το φαινόμενο αυτό είναι ακόμα πιο έντονο.

4.4 Εξοικονόμηση καυσίμου

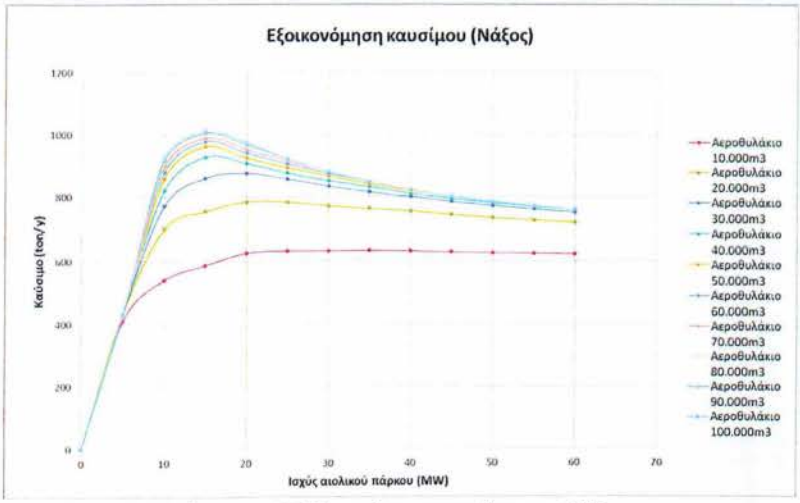
Σε αυτή την ομάδα διαγραμμάτων παρατίθεται η εξοικονόμηση καυσίμου λόγω της χρήσης του Wind-CAES σε σύγκριση με την χρήση αιολικού πάρκου και συμβατικής μονάδας κύκλου Brayton.



Διάγραμμα 28: Εξοικονόμηση καυσίμου στην Άνδρο

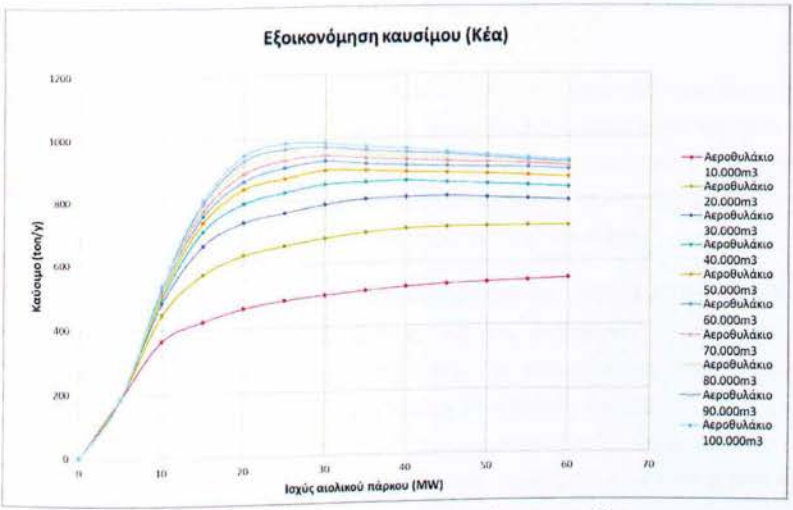


Διάγραμμα 29: Εξοικονόμηση καυσίμου στην Κρήτη

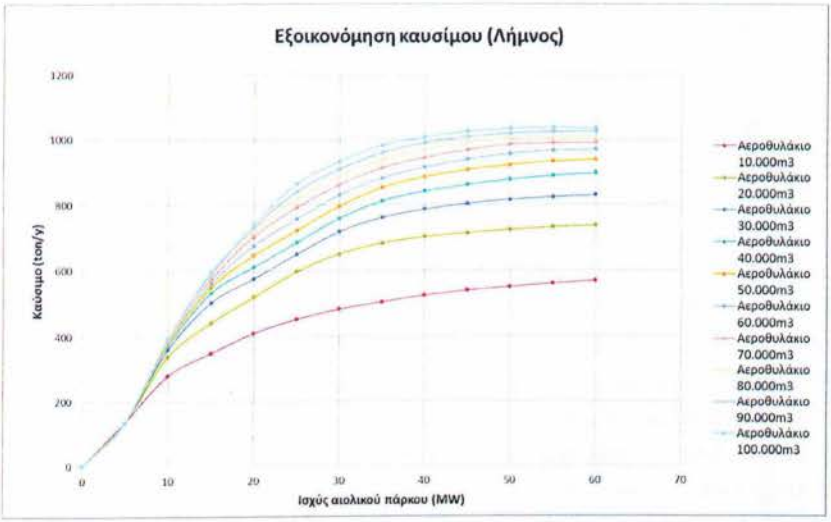


Διάγραμμα 30: Εξοικονόμηση καυσίμου στη Νάξο

Από τα παραπάνω διαγράμματα των περιοχών με υψηλό αιολικό δυναμικό παρατηρούμε ότι ακόμα και με αεροθυλάκιο όγκου 10.000m^3 είναι δυνατόν να επιτευχθεί αξιοσημείωτη οικονομία φυσικού αερίου της τάξης των 500-600 τόνων ανά έτος. Αυτό είναι πολύ σημαντικό, μιας και δίνει την δυνατότητα αποθήκευσης σε δοχεία πίεσης και όχι σε φυσικά αεροθυλάκια τα οποία μπορεί να μην είναι διαθέσιμα σε όλες τις περιοχές. Επίσης, από τα διαγράμματα φαίνεται ότι η εξοικονόμηση καυσίμου είναι αντιστρόφως ανάλογη του αιολικού δυναμικού γεγονός αναμενόμενο, μιας και οι περιοχές με υψηλό δυναμικό καλύπτουν μεγάλο μέρος της ζήτησης απευθείας από το αιολικό πάρκο μειώνοντας εξ αρχής την εξάρτησή τους σε ορυκτά καύσιμα.



Διάγραμμα 31: Εξοικονόμηση καυσίμου στην Κέα



Διάγραμμα 32: Εξοικονόμηση καυσίμου στην Λήμνο

Η σημαντική εξοικονόμηση ενέργειας είναι εμφανής και στις περιοχές με χαμηλό αιολικό δυναμικό όπου η χρήση του Dual mode CAES είναι επιβεβλημένη για την κάλυψη της ζήτησης. Η εξοικονόμηση καυσίμου κυμαίνεται από 400 έως 1000 τόνους καυσίμου ανά έτος.

5 Συμπεράσματα

Η χρήση του λογισμικού "Wind-CAES" δίνει την δυνατότητα μελέτης της ενεργειακής αυτονομίας διαφόρων περιοχών όπου σχεδιάζεται εγκατάσταση αιολικού πάρκου και διάταξης αποθήκευσης ενέργειας (CAES) αλλά και διπλής λειτουργίας (Dual-CAES).

Η παραπάνω μελέτη μας έδωσε την δυνατότητα να υπολογίζουμε όλους τους πιθανούς συνδυασμούς αιολικού πάρκου – αποθήκευσης αέρα ώστε να επιτευχθεί ενεργειακή αυτονομία στην υπό μελέτη περιοχή. Με το "Wind-CAES" παρέχεται η δυνατότητα μεταβολής των μεταβλητών που επηρεάζουν την παραγωγή ενέργειας και έτσι ελέγχονται οι κατασκευαστικές παράμετροι της μονάδας.

Τα αποτελέσματα της παραπάνω μελέτης κατέδειξαν ότι το αιολικό δυναμικό κάθε περιοχής αποτελεί καθοριστικό παράγοντα για την ενεργειακή αυτονόμηση της. Όπως φαίνεται στα παραπάνω διαγράμματα, οι περιοχές με υψηλό αιολικό δυναμικό αυτονομούνται με την χρήση αιολικού πάρκου και απλού CAES, ενώ οι περιοχές με χαμηλό δυναμικό απαιτούν την λειτουργία της μονάδας Dual-CAES. Επίσης, είναι εμφανής η σημαντική εξοικονόμηση καυσίμου που επιτυγχάνεται σε όλες τις περιοχές ανεξαρτήτου του αιολικού τους δυναμικού. Η εξοικονόμηση

φυσικών πόρων δημιουργεί σημαντικά οικονομικά αλλά και περιβαλλοντικά οφέλη στις τοπικές κοινότητες αλλά και στο σύνολο.

Ο συνδυασμός της παραγωγής ενέργειας μέσω ανανεώσιμων πηγών ενέργειας και η αποθήκευση της ενισχύουν σημαντικά την προσπάθεια αύξησης της διείσδυσης των ΑΠΕ και ταυτόχρονα απαλείφουν αρκετά από τα προβλήματα που δημιουργεί η παραγωγή ενέργειας μέσω ΑΠΕ. Από την άλλη πλευρά, η χωροθέτησή εγκαταστάσεων αποθήκευσης του συμπιεσμένου αέρα αλλά και φυσικού αερίου σε νησιωτικές περιοχές είναι αρκετά πολύπλοκη διαδικασία. Ειδικότερα, ιδιαίτερα δύσκολη είναι η εύρεση φυσικού θύλακα που να μπορεί να χρησιμοποιηθεί για την αποθήκευση συμπιεσμένου αέρα αλλά και η κατασκευή μονάδας αποθήκευσης υγροποιημένου φυσικού αερίου. Εναλλακτικά, για τον συμπιεσμένο αέρα μπορούν να χρησιμοποιηθούν δοχεία πίεσης, τα οποία όμως περιορίζουν την δυναμικότητα της μονάδας και θα πρέπει να χωροθετηθούν σε περιοχές όπου δεν θα επηρεάζουν το φυσικό τοπίο νησιωτικών περιοχών. Όσο αφορά την αποθήκευση του φυσικού αερίου, η μόνη λύση που μπορεί να εφαρμοστεί είναι αυτή της διασύνδεσης των νησιών με το δίκτυο παρακείμενων περιοχών. Αντίθετα με τις υπόλοιπες νησιωτικές περιοχές, στην περιοχή της Κρήτης υπάρχει η δυνατότητα δημιουργίας μιας μονάδας CAES, λόγω του μεγέθους του νησιού αλλά και των σχεδιαζόμενων υποδομών του.

6 Βιβλιογραφία

1. Δικτυακός τόπος Energy Storage Association *www.energystorage.org*
2. Stanley Atcitty, *"Electrochemical Capacitor Characterization for Electric Utility Applications"*,Phd, Dissertation, November 2006, Virginia Polytechnic Institute.
3. A.Ter-Gazarian,*"Energy storage for Power systems"*,Peter Peregrinus IEE Energy Series 6,UK, 1994
4. S. Swaminathan, R.K. Sen, *"Review of Power Quality Applications of Energy Storage Systems"*,Sandia National Laboratories, report no. SAND98-1513, July 1998.
   http://www.prod.sandia.gov/cgi-bin/techlib/access-control.pl/1998/981513.pdf
5. Melissa M. Reading, *"Flywheel Energy Storage System", www.energy.ca.gov/reports/2004-04-07_500- 04-014.PDF*
6. S. van der Linden *"Bulk energy storage potential in the USA, current developments and future prospects"*, J. Energy 31 (2006) pp 3446–3457
7. 26. EU-DEEP *"The birth of a EUropean Distributed EnErgy Partnership that will help the large-scale implementation of distributed energy resources in Europe."*FP6 Project: SES6-CT-2003 503516,Technical Annex, available *www.eu-deep.com*
8. 45 A.Tsikalakis, N.Hatziargyriou, J.Dam et al. *"Energy resource planning and control tool for Multi- Users Parks"* in Proc of the, Med Power 2002 Conference, MED02-310 Athens, November 2002
9. 46 ENSURE: *"ENergy Services for multi-User business parks with enhanced levels of REnewables and Rational Use of Energy"*, Description of work of Project NNE5 395 2001, ENERGIE4-T1, EC V Framework Programme,.
10. *"Νέες προοπτικές για την Ανάπτυξη της αντλησιοταμίευσης στην Ελλάδα Τεχνολογία και Τεχνολογικοί Περιορισμοί"* Δημήτριος Παπαντωνης, ΤΕΕ Ιωάννινα 2009
11. *"Wind Turbine Grid Connection and Interaction"*, Deutsches Windenergie-Institut, Tech Wise A/S, DM Energy
12. *"Huntorf CAES: More than 20 Years of Successful Operation"* by Fritz Crotogino KBB GmbH, Hannover, Germany and Klaus-Uwe Mohmeyer and Dr. Roland Scharf E.ON Kraftwerke Bremen, Germany
13. *"COMPRESSED AIR ENERGY STORAGE (CAES)"*, Dresser Rand™
14. *"The Utsira Wind-Hydrogen Project, Presentation on behalf of Hydro"* Trygve Riis,
15. Διαδικτυακός τόπος
    http://www.rwe.com/web/cms/en/365478/rwe/innovation/projects technologies/energy-storage/project-adele/
16. Ackermann, T., Andersson, G., Söder, L., Distributed generation: a definition. *"Electric Power Systems Research"* 57(3), pp. 195-204, 2001.

17. Kaldellis, J.K., Zafirakis, D., *"Present situation and future prospects of electricity generation in Aegean Archipelago islands."* Energy Policy 35(9), pp. 4623-4639, 2007.
18. Kaldellis, J.K., Zafirakis, D., Kavadias, K., *"Techno-economic comparison of energy storage systems for island autonomous electrical networks. Renewable and Sustainable Energy Reviews"* 13(2), pp. 378-392, 2009.
19. Denholm, P., Kulcinski, G.L., *"Life cycle energy requirements and greenhouse gas emissions from large scale energy storage systems. Energy Conversion and Management"* 45(13-14), pp. 2153-2172, 2004.
20. Denholm, P., *"Improving the technical, environmental and social performance of wind energy systems using biomass-based energy storage. Renewable Energy"* 31(9), pp. 1355-1370, 2006.
21. Bullough, C., Gatzen, C., Jakiel, C., Koller, M., Nowi, A., Zunft, S., *"Advanced adiabatic compressed air energy storage for the integration of wind energy"*, Proceedings of the European Wind Energy Conference, London, UK, 2004.
22. Dayan, A., Flesh, J., Saltiel, C., *"Drying of a porous spherical rock for compressed air energy storage. International Journal of Heat Mass Transfer"* 47(19-20), pp. 4459-4468, 2004.
23. Greenblatt, J.B., Succar, S., Denkenberger, D.C., Williams, R.H., Socolow, R.H., Baseload *"wind energy: modeling the competition between gas turbines and compressed air energy storage for supplemental generation."* Energy Policy 35(3), pp. 1474-1492, 2007.
24. Lund, H., Salgi, G., Elmegaard, B., Andersen A.N., *"Optimal operation strategies of compressed air energy storage (CAES) on electricity spot markets with fluctuating prices."* Applied Thermal Engineering 29(5-6), pp. 799-806, 2009.
25. Cavallo, A.J., *"Controllable and affordable utility-scale electricity from intermittent wind resources and compressed air energy storage (CAES)."* Energy 32(2), pp. 120-127, 2007.
26. Salgi, G., Lund, H., *"System behaviour of compressed-air energy-storage in Denmark with a high penetration of renewable energy sources."* Applied Energy 85(4), pp. 182-189, 2008.

7 Παράρτημα

Οι μεταβλητές που χρησιμοποιήθηκαν στον αλγόριθμο είναι παραθέτονται στον παρακάτω πίνακα:

Μέγεθος	Συμβολισμός	Πρόγραμμα
Ταχύτητα αέρα	V (m/s)	Wind_Speed
γ-1/γ		G_Power
Ma*λ		MAL
Βάθος εκφόρτισης	DOD (%)	DOD
Ισχύς Α/Γ	N_{wt} (MW)	Wind_Turbine_Power
Ισχύς Συμπιεστή	N_{com} (MW)	Compressor_Power
Ζητούμενη Ενέργεια	E_{res}(MWh)	Energy_Need_List
Ισχύς Προς Αποθήκευση	N_{stor} (MW)	Power_Storage
Ισχύς Εξόδου	N_{ex}(MW)	DUAL_Turbine_Power_List
Έλλειμμα Ενέργειας	E_{need} (MWh)	Final_Energy_Need_List
Απορρίψεις Ενέργειας	E_{rej} (MWh)	Energy_Rejection_List
Κατανάλωση Ενέργειας	E_{cons} (MWh)	Energy_Consumption_List
Β.Α. Συμπιεστή	H_{com} (%)	CP_Compressor
Β.Α. Στροβίλου	H_{tur} (%)	CP_Turbine
Β.Α. Brayton	H_{bra} (%)	CP_Brayton
Λόγος Συμπίεσης	Πc	Compressor_Ratio
Λόγος Εκτόνωσης	Πt	Expansion_Ratio
Όγκος Αεροθαλάμου	V_{stor} (m3)	Air_Storage_Volume
Αρ. Κατοίκων	No_{res}	Residents
Στοιχειομετρική αναλογία αέρα		Stoichiometric_Ratio
Αναλογία αέρα-καυσίμου	λ	Lamda
Θερμογόνος δύναμη	Hu(MJ/kg)	HU
Θερμοκρασία θαλάμου καύσης	T_{cc} (C)	Combustion_Chamber_Temperature
Θερμοκρασία Αεροθυλακίου	T_{stor} (C)	Storage_Temperature
Θερμοχωρητικότητα καυσαερίων	Cp_{gas} (kJ/kgK)	Gas_Heat_Capacity
Θερμοχωρητικότητα αέρα	Cp_{air} (Kj/kgK)	Air_Heat_Capacity
Παροχή Αέρα Προς Συμπίεση	Ma_{com} (kg/s)	Mass_Air_Compressor_List
Απαιτούμενη μάζα καυσαερίων	Mg_{dem} (kg/s)	CAES_Mass_Gas_List

Απαιτούμενη μάζα αέρα	Ma_{dem} (kg/s)	CAES_Mass_Air_List
Απαιτούμενη μάζα καυσίμου	Mf (kg/s)	CAES_Mass_Fuel_List
Στάθμη αεροθαλάμου	$Level_{stor}$	Air_Storage_Level_List
Διαφορά στάθμης αεροθυλακίου	$Diff_{stor}$	Storage_Difference_List
Πραγματική μάζα αέρα	Ma_{act} (kg/s)	MA_Actual_List
Ώρες μερικής η μηδενικής κάλυψης	No_{caes}	CAES_Non_Coverage_Hours
Περίσσεια Πάρκου		
Μάζα καυσίμου CAES	Mf_{caes} (kg)	Annual_CAES_Mass_Fuel
Μάζα καύσιμου DUAL	Mf_{dual} (kg)	Annual_DUAL_Mass_fuel
Μάζα καύσιμου BRAYTON	Mf_{bray} (kg)	Annual_BRAYTON_Mass_Fuel
Συνολική Μάζα καύσιμου	M_{fuel} (kg)	Annual_Total_Mass_Fuel
Εξοικονόμηση καυσίμου	ΔMf (tons/year)	DMf
Νηνεμία	No_{wind}	Wind_Calms
Μέγιστη μάζα αποθ. αέρα	Mmax (kg air)	
Ελάχιστη μάζα αποθ. αέρα	Mmin (kg air)	

Πίνακας 6: Αντιστοιχία μεταβλητών-μεγεθών αλγορίθμου

7.1 Τυπολόγιο

1. Για την κατανάλωση ενέργειας ανά άτομο, διαιρούμε τον αριθμό των κατοίκων (αναγωγή ανά 4μελή οικογένεια) και το πολλαπλασιάζουμε με ενεργειακό μας προφίλ:

$$E_{cons} = \frac{No_{res}}{4} \cdot E_{fam} \cdot 10^{-6}$$

(Μόνο εφόσον επιλεγεί η χρήση του προεπιλεγμένου προφίλ κατανάλωσης)

2. Η ζητούμενη ενέργεια, ορίζεται ως η διαφορά της παραγόμενης με την καταναλισκόμενη:

$$E_{res} = E_{prod} - E_{cons}$$

3. Η αποθηκευμένη ενέργεια υπολογίζεται ως η ζητούμενη επί το βαθμό απόδοσης του συμπιεστή:

$$E_{stor} = E_{res}.\eta_{com}$$

4. Η μάζα του αέρα προς συμπίεση είναι το γινόμενο της αποθηκευμένης ενέργειας επί το βαθμό απόδοσης του συμπιεστή προς τη θερμοχωρητικότητα του αέρα, τη θερμοκρασία και το λόγο συμπίεσης:

$$Ma_{com} = \frac{E_{stor}.1000}{Cp_{air}.T.\left(\Pi c^{\frac{\gamma-1}{\gamma}} - 1\right)}.\eta_{com}$$

5. Αντίστοιχα ορίζεται και η απαιτούμενη μάζα των καυσαερίων:

$$Mg_{dem} = -\frac{E_{res}.1000}{Cp_{gas}.T.\left(1 - \frac{1}{\Pi\tau^{\frac{\gamma-1}{\gamma}}}\right)}.\eta_{tur}$$

6. Η απαιτούμενη μάζα του αέρα, υπολογίζεται μέσω των καυσαερίων, του γινομένου της μάζας του αέρα και του "λ" αλλά και της θερμογόνου δύναμης του καυσίμου:

$$Ma_{dem} = \frac{Mg_{dem}.Cp_{gas}.T}{\frac{Cp_{air}.T_{stor} + Hu.1000}{Ma.\lambda}}$$

7. Όσον αφορά τη μάζα του καυσίμου που απαιτείται για την κάλυψη της ζήτησης, αφαιρούμε τη μάζα του αέρα από τη μάζα των παραγόμενων κατά την καύση καυσαερίων:

$$M_f = M_{gas} - M_{air}$$

8. Η στάθμη του αεροθαλάμου ορίζεται ως το άθροισμα του λόγου αέρα συμπίεσης συν την ελάχιστη χωρητικότητα για την πρώτη τιμή:

$$Level_{stor}(i) = Ma_{com} + Ma_{min}$$

9. Ενώ για τις υπόλοιπες τιμές είναι η απαιτούμενη μάζα επί 3600:

$$Level_{stor}(i+1) = Ma_{dem}.3600$$

10. Η μεταβολή της στάθμης υπολογίζεται ως η επόμενη μείον την προηγούμενη τιμή:

$$Diff_{stor} = Level_{stor}(i+1) - Level_{stor}(i)$$

11. Η διαθέσιμη μάζα του αέρα δίνεται παρακάτω:

$$Ma_{act} = Diff_{stor}.3600$$

12. Η ισχύς εξόδου του συμπιεστή δίνεται παρακάτω:

$$N_{ex} = Ma_{act}.\left(1+\frac{1}{MAL}\right).T.Cp_{gas}.\left(1-\frac{1}{\Pi\tau^{\frac{\gamma-1}{\gamma}}}\right).10^{-3}.\eta_{tur}$$

13. Οι απορρίψεις ενέργειας δίνονται από τη διαφορά παραγόμενης και καταναλισκόμενης ενέργειας:

$$E_{rej} = E_{prod} - E_{cons}$$

ΑΠΟΤΕΛΕΣΜΑΤΑ:

1. Το τελικό έλλειμμα ενέργειας δίνεται από το άθροισμα των ωριαίων ζητούμενων ενεργειών του έτους:

$$E_{need} = \sum_{8764}^{i=1} E_{res}(i)$$

2. Αντίστοιχα και το άθροισμα των απορρίψεων:

$$E_{REJ} = \sum_{8764}^{i=1} E_{rej}(i)$$

3. Η ειδική κατανάλωση καυσίμου:

$$Heat\ Rate = \frac{Mf_{CAES}.HU}{3{,}6.\sum N_{ex}}$$

4. Η συνολική μάζα του καυσίμου για τον κύκλο Brayton δίνεται:

$$Mf_{BRAY} = \sum_{8764}^{i=1} N_{res}(i)\,.\,3{,}6.\frac{1}{HU.\eta_{bray}}$$

5. Η συνολική μάζα του καυσίμου για το CAES:

$$Mf_{CAES} = \sum_{8764}^{i=1} Ma_{act} \,.\, 3{,}6 . \frac{1}{MAL}$$

6. Ενώ η μάζα καυσίμου για το DUAL MODE υπολογίζετε ως:

$$Mf_{DUAL} = \frac{E_{need} . 3{,}6}{HU . \eta_{bray}}$$

7. Τέλος η συνολική κατανάλωση καυσίμου είναι το άθροισμα των παραπάνω:

$$Mf_{TOTAL} = Mf_{CAES} + Mf_{DUAL}$$

8. Ενώ η διαφορά μάζας καυσίμου είναι:

$$\Delta Mf = Mf_{TOTAL} - Mf_{BRAY}$$

9. Η παραγωγή ενέργειας της ανεμογεννήτριας υπολογίζεται από την ακόλουθη κλαδική συνάρτηση. Όπου v η ταχύτητα του ανέμου σε m/s.

$$N_{WT}(v) = \left\{ \begin{array}{ll} 0 & v < 4{,}0 \\ N_{tur} . (0{,}0126\, v^2 - 0{,}072\, v + 0{,}1177) & 4{,}0 < v < 8{,}0 \\ N_{tur} . (0{,}1546\, v - 0{,}8811) & \gamma\iota\alpha \quad 8{,}0 < v < 11{,}5 \\ N_{tur} (-0{,}0206\, v^2 + 0{,}5621 . v - 2{,}8397) & 11{,}5 < v < 14{,}0 \\ N_{tur} & 14{,}0 < v < 25{,}0 \\ 0 & v > 25{,}0 \end{array} \right\}$$

7.2 Ο αλγόριθμος

Ο αλγόριθμος αναπτύχθηκε στην γλώσσα προγραμματισμού C# με την χρήση του λογισμικού Visual Studio.

7.2.1 Αλγόριθμος περιβάλλοντος εργασίας

```
<MetroControls:MetroWindow
    xmlns="http://schemas.microsoft.com/winfx/2006/xaml/presentation"
    xmlns:x="http://schemas.microsoft.com/winfx/2006/xaml"
    xmlns:i="http://schemas.microsoft.com/expression/2010/interactivity"
    xmlns:Metro="clr-namespace:MahApps.Metro;assembly=MahApps.Metro"
    xmlns:MetroControls="clr-
namespace:MahApps.Metro.Controls;assembly=MahApps.Metro"
    xmlns:MetroBehaviours="clr-
namespace:MahApps.Metro.Behaviours;assembly=MahApps.Metro"
    xmlns:MetroConverters="clr-
namespace:MahApps.Metro.Converters;assembly=MahApps.Metro"
    xmlns:d="http://schemas.microsoft.com/expression/blend/2008"
xmlns:mc="http://schemas.openxmlformats.org/markup-compatibility/2006"
mc:Ignorable="d" x:Class="CAES.MainWindow"
    Title="CAES Study" Height="480" Width="640" ResizeMode="NoResize"
Icon="/CAES;component/Resources/windturbinemetro.png">

  <MetroControls:MetroWindow.Resources>
    <ResourceDictionary>
      <ResourceDictionary.MergedDictionaries>
        <ResourceDictionary
Source="pack://application:,,,/MahApps.Metro;component/Styles/Colours.xaml"/>
        <ResourceDictionary
Source="pack://application:,,,/MahApps.Metro;component/Styles/Fonts.xaml"/>
        <ResourceDictionary
Source="pack://application:,,,/MahApps.Metro;component/Styles/Controls.xaml"/>
        <ResourceDictionary
Source="pack://application:,,,/MahApps.Metro;component/Styles/Accents/Blue.xa
ml"/>
        <ResourceDictionary
Source="pack://application:,,,/MahApps.Metro;component/Styles/Accents/BaseLigh
t.xaml"/>
      </ResourceDictionary.MergedDictionaries>
      <Storyboard x:Key="StartIn">
          <DoubleAnimationUsingKeyFrames
Storyboard.TargetProperty="(UIElement.Opacity)" Storyboard.TargetName="Start">
              <EasingDoubleKeyFrame KeyTime="0" Value="0"/>
              <EasingDoubleKeyFrame KeyTime="0:0:0.5" Value="1"/>
          </DoubleAnimationUsingKeyFrames>
```

```
            <ObjectAnimationUsingKeyFrames
Storyboard.TargetProperty="(UIElement.Visibility)" Storyboard.TargetName="Start">
                <DiscreteObjectKeyFrame KeyTime="0" Value="{x:Static
Visibility.Visible}"/>
                <DiscreteObjectKeyFrame KeyTime="0:0:0.5" Value="{x:Static
Visibility.Visible}"/>
            </ObjectAnimationUsingKeyFrames>
        </Storyboard>
        <Storyboard x:Key="StartOut">
            <DoubleAnimationUsingKeyFrames
Storyboard.TargetProperty="(UIElement.Opacity)" Storyboard.TargetName="Start">
                <EasingDoubleKeyFrame KeyTime="0" Value="1"/>
                <EasingDoubleKeyFrame KeyTime="0:0:0.5" Value="0"/>
            </DoubleAnimationUsingKeyFrames>
            <ObjectAnimationUsingKeyFrames
Storyboard.TargetProperty="(UIElement.Visibility)" Storyboard.TargetName="Start">
                <DiscreteObjectKeyFrame KeyTime="0" Value="{x:Static
Visibility.Visible}"/>
                <DiscreteObjectKeyFrame KeyTime="0:0:0.5" Value="{x:Static
Visibility.Collapsed}"/>
            </ObjectAnimationUsingKeyFrames>
        </Storyboard>
        <Storyboard x:Key="Step1In">
            <DoubleAnimationUsingKeyFrames
Storyboard.TargetProperty="(UIElement.Opacity)"
Storyboard.TargetName="Step1">
                <EasingDoubleKeyFrame KeyTime="0" Value="0"/>
                <EasingDoubleKeyFrame KeyTime="0:0:0.5" Value="1"/>
            </DoubleAnimationUsingKeyFrames>
            <ObjectAnimationUsingKeyFrames
Storyboard.TargetProperty="(UIElement.Visibility)"
Storyboard.TargetName="Step1">
                <DiscreteObjectKeyFrame KeyTime="0" Value="{x:Static
Visibility.Visible}"/>
                <DiscreteObjectKeyFrame KeyTime="0:0:0.5" Value="{x:Static
Visibility.Visible}"/>
            </ObjectAnimationUsingKeyFrames>
        </Storyboard>
        <Storyboard x:Key="Step1Out">
            <DoubleAnimationUsingKeyFrames
Storyboard.TargetProperty="(UIElement.Opacity)"
Storyboard.TargetName="Step1">
                <EasingDoubleKeyFrame KeyTime="0" Value="1"/>
                <EasingDoubleKeyFrame KeyTime="0:0:0.5" Value="0"/>
            </DoubleAnimationUsingKeyFrames>
```

```
            <ObjectAnimationUsingKeyFrames
Storyboard.TargetProperty="(UIElement.Visibility)"
Storyboard.TargetName="Step1">
                <DiscreteObjectKeyFrame KeyTime="0" Value="{x:Static
Visibility.Visible}"/>
                <DiscreteObjectKeyFrame KeyTime="0:0:0.5" Value="{x:Static
Visibility.Collapsed}"/>
            </ObjectAnimationUsingKeyFrames>
        </Storyboard>
        <Storyboard x:Key="Step2In">
            <DoubleAnimationUsingKeyFrames
Storyboard.TargetProperty="(UIElement.Opacity)"
Storyboard.TargetName="Step2">
                <EasingDoubleKeyFrame KeyTime="0" Value="0"/>
                <EasingDoubleKeyFrame KeyTime="0:0:0.5" Value="1"/>
            </DoubleAnimationUsingKeyFrames>
            <ObjectAnimationUsingKeyFrames
Storyboard.TargetProperty="(UIElement.Visibility)"
Storyboard.TargetName="Step2">
                <DiscreteObjectKeyFrame KeyTime="0" Value="{x:Static
Visibility.Visible}"/>
                <DiscreteObjectKeyFrame KeyTime="0:0:0.5" Value="{x:Static
Visibility.Visible}"/>
            </ObjectAnimationUsingKeyFrames>
        </Storyboard>
        <Storyboard x:Key="Step2Out">
            <DoubleAnimationUsingKeyFrames
Storyboard.TargetProperty="(UIElement.Opacity)"
Storyboard.TargetName="Step2">
                <EasingDoubleKeyFrame KeyTime="0" Value="1"/>
                <EasingDoubleKeyFrame KeyTime="0:0:0.5" Value="0"/>
            </DoubleAnimationUsingKeyFrames>
            <ObjectAnimationUsingKeyFrames
Storyboard.TargetProperty="(UIElement.Visibility)"
Storyboard.TargetName="Step2">
                <DiscreteObjectKeyFrame KeyTime="0" Value="{x:Static
Visibility.Visible}"/>
                <DiscreteObjectKeyFrame KeyTime="0:0:0.5" Value="{x:Static
Visibility.Collapsed}"/>
            </ObjectAnimationUsingKeyFrames>
        </Storyboard>
        <Storyboard x:Key="Step3In">
            <DoubleAnimationUsingKeyFrames
Storyboard.TargetProperty="(UIElement.Opacity)"
Storyboard.TargetName="Step3">
                <EasingDoubleKeyFrame KeyTime="0" Value="0"/>
```

```
                    <EasingDoubleKeyFrame KeyTime="0:0:0.5" Value="1"/>
                </DoubleAnimationUsingKeyFrames>
                <ObjectAnimationUsingKeyFrames
Storyboard.TargetProperty="(UIElement.Visibility)"
Storyboard.TargetName="Step3">
                    <DiscreteObjectKeyFrame KeyTime="0" Value="{x:Static
Visibility.Visible}"/>
                    <DiscreteObjectKeyFrame KeyTime="0:0:0.5" Value="{x:Static
Visibility.Visible}"/>
                </ObjectAnimationUsingKeyFrames>
            </Storyboard>
            <Storyboard x:Key="Step3Out">
                <DoubleAnimationUsingKeyFrames
Storyboard.TargetProperty="(UIElement.Opacity)"
Storyboard.TargetName="Step3">
                    <EasingDoubleKeyFrame KeyTime="0" Value="1"/>
                    <EasingDoubleKeyFrame KeyTime="0:0:0.5" Value="0"/>
                </DoubleAnimationUsingKeyFrames>
                <ObjectAnimationUsingKeyFrames
Storyboard.TargetProperty="(UIElement.Visibility)"
Storyboard.TargetName="Step3">
                    <DiscreteObjectKeyFrame KeyTime="0" Value="{x:Static
Visibility.Visible}"/>
                    <DiscreteObjectKeyFrame KeyTime="0:0:0.5" Value="{x:Static
Visibility.Collapsed}"/>
                </ObjectAnimationUsingKeyFrames>
            </Storyboard>
            <Storyboard x:Key="FinishIn">
                <DoubleAnimationUsingKeyFrames
Storyboard.TargetProperty="(UIElement.Opacity)"
Storyboard.TargetName="Finish">
                    <EasingDoubleKeyFrame KeyTime="0" Value="0"/>
                    <EasingDoubleKeyFrame KeyTime="0:0:0.5" Value="1"/>
                </DoubleAnimationUsingKeyFrames>
                <ObjectAnimationUsingKeyFrames
Storyboard.TargetProperty="(UIElement.Visibility)"
Storyboard.TargetName="Finish">
                    <DiscreteObjectKeyFrame KeyTime="0" Value="{x:Static
Visibility.Visible}"/>
                    <DiscreteObjectKeyFrame KeyTime="0:0:0.5" Value="{x:Static
Visibility.Visible}"/>
                </ObjectAnimationUsingKeyFrames>
            </Storyboard>
            <Storyboard x:Key="FinishOut">
```

```
                <DoubleAnimationUsingKeyFrames
Storyboard.TargetProperty="(UIElement.Opacity)"
Storyboard.TargetName="Finish">
                    <EasingDoubleKeyFrame KeyTime="0" Value="1"/>
                    <EasingDoubleKeyFrame KeyTime="0:0:0.5" Value="0"/>
                </DoubleAnimationUsingKeyFrames>
                <ObjectAnimationUsingKeyFrames
Storyboard.TargetProperty="(UIElement.Visibility)"
Storyboard.TargetName="Finish">
                    <DiscreteObjectKeyFrame KeyTime="0" Value="{x:Static
Visibility.Visible}"/>
                    <DiscreteObjectKeyFrame KeyTime="0:0:0.5" Value="{x:Static
Visibility.Collapsed}"/>
                </ObjectAnimationUsingKeyFrames>
            </Storyboard>
        </ResourceDictionary>
    </MetroControls:MetroWindow.Resources>
        <MetroControls:MetroWindow.Triggers>
            <EventTrigger RoutedEvent="FrameworkElement.Loaded">
                <BeginStoryboard Storyboard="{StaticResource StartIn}"/>
            </EventTrigger>
        </MetroControls:MetroWindow.Triggers>

    <i:Interaction.Behaviors>
        <MetroBehaviours:BorderlessWindowBehavior ResizeWithGrip="True" />
    </i:Interaction.Behaviors>

    <Grid x:Name="PageHolder">
        <!--<Grid.Background>
            <RadialGradientBrush Center="0.504,0.83" GradientOrigin="0.504,0.83"
RadiusY="0.891" RadiusX="0.573">
                <GradientStop Color="#FF161616" Offset="0.992"/>
                <GradientStop Color="#FF5C5C5C"/>
            </RadialGradientBrush>
        </Grid.Background>-->
        <Grid x:Name="Start" Opacity="0" Visibility="Collapsed">
            <TextBlock Margin="25,25,25,55" x:Name="CAESDetailslbl" Text="The CAES
Study program is an engineering tool for energy study in Compressed Air Energy
Storage plants combined with Wind Farms." FontSize="24" FontFamily="Segoe UI"
TextWrapping="Wrap" />
            <Button Content="START" Margin="0,0,25,25" x:Name="StartStep1btn"
Height="25" VerticalAlignment="Bottom" HorizontalAlignment="Right" Width="75"
Click="StartStep1btn_Click" />
        </Grid>
        <Grid x:Name="Step1" Opacity="0" Visibility="Collapsed">
```

```
        <Label Content="step 1 - import data from excel" Height="54"
Margin="25,25,25,0" x:Name="CAESSubtitlelbl_Step1" VerticalAlignment="Top"
FontSize="32" FontFamily="Segoe UI" />
        <TextBlock Margin="25,88,25,0" x:Name="CAESDetailslbl_Step1"
FontSize="18.667" FontFamily="Segoe UI" TextWrapping="Wrap" Height="113.882"
VerticalAlignment="Top" ><Run Text="Please select the excel file which contains the
following:"/><LineBreak/><Run Text="- Wind Speed"/><LineBreak/><Run Text="-
Temperature (if available)"/><LineBreak/><Run Text="- Energy Demand (if
available)"/></TextBlock>
        <TextBox x:Name="ExcelFilePathTxtBox" Margin="25,205.882,249,220.118"
TextWrapping="Wrap" IsEnabled="False" d:LayoutOverrides="Height"/>
        <Button x:Name="BrowseExcelFileBtn" Content="BROWSE"
HorizontalAlignment="Right" Margin="0,205.882,138.672,220.118" Width="90.376"
Click="BrowseExcelFileBtn_Click"/>
        <CheckBox x:Name="DefaultEnergyDemandCheckBox" Content="Use Default
Energy Demand Profile" HorizontalAlignment="Left" Margin="25,0,0,163"
VerticalAlignment="Bottom" Checked="CheckBox_Checked"
Unchecked="CheckBox_Unchecked"/>
        <TextBox x:Name="ResidentsTxtBox" Margin="25,0,0,63.131"
TextWrapping="Wrap" VerticalAlignment="Bottom" ToolTip="Number of Residents"
FontSize="14.667" HorizontalAlignment="Left" Width="120.317" Height="27.631"
Visibility="Collapsed"/>
        <Label x:Name="Residentslbl" Content="Residents" Height="34.539"
Margin="19.818,0,0,94.762" VerticalAlignment="Bottom"
HorizontalAlignment="Left" Width="148.516" FontSize="18.667"
Visibility="Collapsed"/>
        <Button Content="NEXT" Margin="0,0,25,25" x:Name="StartStep2btn"
Height="25" VerticalAlignment="Bottom" HorizontalAlignment="Right" Width="75"
Click="StartStep2btn_Click" d:LayoutOverrides="HorizontalAlignment,
VerticalAlignment" />
        <CheckBox x:Name="DefaultTemperatureListCheckBox" Content="Use
Default Temperature List" HorizontalAlignment="Left" Margin="25,0,0,193"
VerticalAlignment="Bottom" Unchecked="CheckBox_Unchecked"/>
    </Grid>
    <Grid x:Name="Step2" Opacity="0" Visibility="Collapsed">
        <Label Content="step 2 - input variables" Height="54" Margin="25,25,25,0"
x:Name="CAESSubtitlelbl_Step2" VerticalAlignment="Top" FontSize="32"
FontFamily="Segoe UI" />
        <Label Content="HU (MJ/Kg):" Height="42" Margin="61,130.633,0,0"
VerticalAlignment="Top" FontSize="16" HorizontalAlignment="Left" Width="107"
d:LayoutOverrides="VerticalAlignment"/>
        <TextBox x:Name="HUTxtBox" Height="30" Margin="249,133,305,0"
TextWrapping="Wrap" VerticalAlignment="Top" FontSize="18.667" MaxLength="5"
d:LayoutOverrides="VerticalAlignment" />
        <Label Content="Air Ratio (λ):" Height="42" Margin="61,167,0,0"
VerticalAlignment="Top" FontSize="16" HorizontalAlignment="Left" Width="107"/>
```

```
        <TextBox x:Name="LamdaTxtBox" Height="30" Margin="249,172.623,305,0"
TextWrapping="Wrap" VerticalAlignment="Top" FontSize="18.667" MaxLength="5"
d:LayoutOverrides="VerticalAlignment" />
        <Label Content="Cavern Storage Temp. (K):" Height="46"
Margin="33,203.623,0,0" VerticalAlignment="Top" FontSize="16"
HorizontalAlignment="Left" Width="196"/>
        <TextBox x:Name="CavernStorageTempTxtBox" Height="30"
Margin="249,211.623,305,0" TextWrapping="Wrap" VerticalAlignment="Top"
FontSize="18.667" MaxLength="5" />
        <Label Content="Air Heat Capacity (KJ/Kg*K):" Height="42"
Margin="29,241.877,0,0" VerticalAlignment="Top" FontSize="16"
HorizontalAlignment="Left" Width="212"/>
        <TextBox x:Name="AirHeatCapacityTxtBox" Height="30"
Margin="249,252.877,305,0" TextWrapping="Wrap" VerticalAlignment="Top"
FontSize="18.667" MaxLength="5" />
        <Label Content="Gas Heat Capacity (KJ/Kg*K):" Height="46"
Margin="29,279.877,0,0" VerticalAlignment="Top" FontSize="16"
HorizontalAlignment="Left" Width="212"/>
        <TextBox x:Name="GasHeatCapacityTxtBox" Height="30"
Margin="249,286.877,305,0" TextWrapping="Wrap" VerticalAlignment="Top"
FontSize="18.667" MaxLength="5" />
        <Label Content="Brayton Efficiency (0,1]:" Height="42"
Margin="33,318.877,0,0" VerticalAlignment="Top" FontSize="16"
HorizontalAlignment="Left" Width="196"/>
        <TextBox x:Name="CPBraytonTxtBox" Height="30"
Margin="249,324.5,305,0" TextWrapping="Wrap" VerticalAlignment="Top"
FontSize="18.667" MaxLength="5" />
        <Label Content="Gas Turbine" HorizontalAlignment="Right" Height="41"
Margin="0,122,93,0" VerticalAlignment="Top" Width="158" FontSize="24"/>
        <Label Content="Efficiency (0,1]:" HorizontalAlignment="Right" Height="30"
Margin="0,167,134,0" VerticalAlignment="Top" Width="117" FontSize="16"/>
        <TextBox x:Name="GasTurbineEfficiencyTxtBox" Height="30"
Margin="0,167,36,0" TextWrapping="Wrap" VerticalAlignment="Top"
FontSize="18.667" HorizontalAlignment="Right" Width="80" MaxLength="5" />
        <Label Content="Pressure Ratio:" HorizontalAlignment="Right"
Margin="0,203.623,134,208.377" Width="117" FontSize="16"/>
        <TextBox x:Name="GasTurbinePressureRatioTxtBox"
Margin="0,203.623,36,0" TextWrapping="Wrap" FontSize="18.667"
HorizontalAlignment="Right" Width="80" Height="30" VerticalAlignment="Top"
MaxLength="5" />
        <Label Content="Compressor" HorizontalAlignment="Right" Height="41"
Margin="0,241.877,93,0" VerticalAlignment="Top" Width="158" FontSize="24"/>
        <Label Content="Efficiency (0,1]:" HorizontalAlignment="Right" Height="30"
Margin="0,286.877,134,0" VerticalAlignment="Top" Width="117" FontSize="16"/>
```

```
        <TextBox x:Name="CompressorEfficiencyTxtBox" Height="30"
Margin="0,286.877,36,0" TextWrapping="Wrap" VerticalAlignment="Top"
FontSize="18.667" HorizontalAlignment="Right" Width="80" MaxLength="5" />
        <Label Content="Pressure Ratio:" HorizontalAlignment="Right"
Margin="0,323.5,134,0" Width="117" FontSize="16" Height="30"
VerticalAlignment="Top" d:LayoutOverrides="Height"/>
        <TextBox x:Name="CompressorPressureRatioTxtBox" Margin="0,324.5,36,0"
TextWrapping="Wrap" FontSize="18.667" Width="80" Height="30"
VerticalAlignment="Top" MaxLength="5" HorizontalAlignment="Right" />
        <Button Content="NEXT" Margin="0,0,25,25" x:Name="StartStep3btn"
Height="25" VerticalAlignment="Bottom" HorizontalAlignment="Right" Width="75"
d:LayoutOverrides="HorizontalAlignment, VerticalAlignment"
Click="StartStep3btn_Click" />
        <Button Content="Back" Margin="0,0,524,25" x:Name="GoToStep1btn"
Height="25" VerticalAlignment="Bottom" HorizontalAlignment="Right" Width="75"
Click="GoToStep1btn_Click" d:LayoutOverrides="VerticalAlignment" />
    </Grid>
    <Grid x:Name="Step3" Opacity="0" Visibility="Collapsed">
        <Label Content="step 3 - input range variables" Height="54"
Margin="25,25,25,0" x:Name="CAESSubtitlelbl_Step3" VerticalAlignment="Top"
FontSize="32" FontFamily="Segoe UI" />
        <Label x:Name="Fromlbl" Content="From" Height="30"
Margin="196,126,0,0" VerticalAlignment="Top" FontSize="18.667"
d:LayoutOverrides="VerticalAlignment" HorizontalAlignment="Left" Width="60"/>
        <Label x:Name="Tolbl" Content="To" Height="30" Margin="0,126,173,0"
VerticalAlignment="Top" FontSize="18.667" HorizontalAlignment="Right"
Width="60" d:LayoutOverrides="VerticalAlignment"/>
        <Label x:Name="Steplbl" Content="Step" Height="40" Margin="0,126,34,0"
VerticalAlignment="Top" FontSize="18.667" HorizontalAlignment="Right"
Width="60" d:LayoutOverrides="VerticalAlignment"/>
        <Label x:Name="WindTurbinePowerlbl" Content="Wind
Turbine&#xd;&#xa;Power (MW):" HorizontalAlignment="Left" Height="65"
Margin="25,163,0,0" VerticalAlignment="Top" Width="139" FontSize="18.667"/>
        <Label x:Name="AirCompressorlbl" Content="Air
Compressor&#xd;&#xa;Power (MW):" HorizontalAlignment="Left" Height="61"
Margin="25,238,0,0" VerticalAlignment="Top" Width="139" FontSize="18.667"/>
        <Label x:Name="AirStoragelbl" Content="Air Storage&#xd;&#xa;Volume
(m³):" HorizontalAlignment="Left" Height="62" Margin="25,312,0,0"
VerticalAlignment="Top" Width="139" FontSize="18.667"/>
        <TextBox x:Name="WindTurbinePowerFromTxtBox" Height="30"
Margin="174,183,0,0" TextWrapping="Wrap" VerticalAlignment="Top"
FontSize="18.667" MaxLength="5" HorizontalAlignment="Left" Width="140"
d:LayoutOverrides="HorizontalAlignment" />
        <TextBox x:Name="WindTurbinePowerToTxtBox" Height="30"
Margin="345,183,0,0" TextWrapping="Wrap" VerticalAlignment="Top"
```

```
FontSize="18.667" HorizontalAlignment="Left" Width="140" MaxLength="5"
RenderTransformOrigin="-0.16,0.7" />
        <TextBox x:Name="WindTurbinePowerStepTxtBox" Height="30"
Margin="534,183,0,0" TextWrapping="Wrap" VerticalAlignment="Top"
FontSize="18.667" HorizontalAlignment="Left" Width="56" MaxLength="5" />
        <TextBox x:Name="AirCompressorFromTxtBox" Margin="174,251,0,0"
TextWrapping="Wrap" FontSize="18.667" MaxLength="5" Height="30"
VerticalAlignment="Top" HorizontalAlignment="Left" Width="140"
d:LayoutOverrides="HorizontalAlignment" />
        <TextBox x:Name="AirCompressorToTxtBox" Margin="345,251,0,0"
TextWrapping="Wrap" FontSize="18.667" HorizontalAlignment="Left" Width="140"
MaxLength="5" Height="30" VerticalAlignment="Top" />
        <TextBox x:Name="AirCompressorStepTxtBox" Margin="534,251,0,0"
TextWrapping="Wrap" FontSize="18.667" HorizontalAlignment="Left" Width="56"
MaxLength="5" Height="30" VerticalAlignment="Top" />
        <TextBox x:Name="AirStorageFromTxtBox" Height="30"
Margin="174,328,0,0" TextWrapping="Wrap" VerticalAlignment="Top"
FontSize="18.667" MaxLength="10" HorizontalAlignment="Left" Width="140"
d:LayoutOverrides="HorizontalAlignment" />
        <TextBox x:Name="AirStorageToTxtBox" Height="30" Margin="345,328,0,0"
TextWrapping="Wrap" VerticalAlignment="Top" FontSize="18.667"
HorizontalAlignment="Left" Width="140" MaxLength="10" />
        <TextBox x:Name="AirStorageStepTxtBox" Height="30"
Margin="513,328,0,0" TextWrapping="Wrap" VerticalAlignment="Top"
FontSize="18.667" HorizontalAlignment="Left" Width="100" MaxLength="5" />
        <Button Content="EXPORT" Margin="0,0,25,25" x:Name="Exportbtn"
Height="25" VerticalAlignment="Bottom" HorizontalAlignment="Right" Width="75"
Click="Exportbtn_Click" d:LayoutOverrides="HorizontalAlignment,
VerticalAlignment" />
        <Button Content="BACK" Margin="0,0,524,25" x:Name="GoToStep2btn"
Height="25" VerticalAlignment="Bottom" HorizontalAlignment="Right" Width="75"
d:LayoutOverrides="VerticalAlignment" Click="GoToStep2btn_Click" />
    </Grid>
    <Grid x:Name="Finish" Opacity="0" Visibility="Collapsed">
        <Label Content="success" Height="54" Margin="25,25,25,0"
x:Name="CAESSubtitlelbl_Finish" VerticalAlignment="Top" FontSize="32"
FontFamily="Segoe UI" />
        <TextBlock Margin="25,121,25,54" x:Name="ExportDetailslbl" FontSize="24"
FontFamily="Segoe UI"  TextWrapping="Wrap" ><Run Text="Results exported to
excel successfully!"/><LineBreak/><Run Text="To start a new study press
&amp;quot;START NEW&amp;quot;"/></TextBlock>
        <Button Content="START NEW" Margin="0,0,25,25"
x:Name="StartNewStudyBtn" Height="25" VerticalAlignment="Bottom"
HorizontalAlignment="Right" Width="75" Click="StartNewStudyBtn_Click"
RenderTransformOrigin="3.907,8.32" d:LayoutOverrides="HorizontalAlignment,
VerticalAlignment" />
```

```
    </Grid>
  </Grid>
</MetroControls:MetroWindow>
```

7.2.2 Αλγόριθμος εισαγωγής και εξαγωγής δεδομένων

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
using System.Windows;
using System.Windows.Controls;
using System.Windows.Data;
using System.Windows.Documents;
using System.Windows.Input;
using System.Windows.Media;
using System.Windows.Media.Imaging;
using System.Windows.Navigation;
using System.Windows.Shapes;
using MahApps.Metro;
using System.Windows.Media.Animation;
using Excel = Microsoft.Office.Interop.Excel;
using Microsoft.Win32;
using System.Windows.Threading; // For Dispatcher.
using System.Globalization;
using System.Threading;

namespace CAES
{
  /// <summary>
  /// Interaction logic for MainWindow.xaml
  /// </summary>
  public partial class MainWindow
  {
    public MainWindow()
    {
      InitializeComponent();
      ThemeManager.ChangeTheme(this, ThemeManager.DefaultAccents.First(a =>
a.Name == "Blue"), Theme.Dark);
      ListConstructors();
    }
```

```
    private static void ListConstructors()
    {
      // Construct all the lists.
      List<double> WindSpeedList = new List<double>();
      List<double> WindTurbinePower = new List<double>();
      List<double> EnergyDemandPerFamily = new List<double>();
      List<double> PowerProductionList = new List<double>();
      List<double> EnergyConsumptionList = new List<double>();
      List<double> EnergyNeedList = new List<double>();
      List<double> EnergyRejectionList = new List<double>();
      List<double> KelvinList = new List<double>();
      List<double> AirHeatCapacityList = new List<double>();
      List<double> PowerStorageList = new List<double>();
      List<double> MassAirCompressorList = new List<double>();
      List<double> CAESMassGasList = new List<double>();
      List<double> CAESMassAirList = new List<double>();
      List<double> CAESMassFuelList = new List<double>();
      List<double> AirStorageLevelList = new List<double>();
      List<double> CAESEnergyNeedList = new List<double>();
      List<double> DualMassFuelList = new List<double>();
    }

    #region Variables

    // Program Variables.
    int currentPage = 1;
    public static string filename;
    public static string defaultExcelFileName =
System.Environment.CurrentDirectory + @"\Excel\DefaultExcel.xls";

    #region Static Values

    public static double AdiabaticCoefficient = 1.4;
    public static double StoichiometricRatio = 15;

    #endregion Static Values

    #region User Input Values

    public static int Residents;

    // Other Variables.
    public static double HU;
    public static double Lamda;
    public static double StorageTemperature;
    public static double AirHeatCapacity;
```

```
public static double GasHeatCapacity;
public static double CPBrayton;

// Compressor Variables.
public static double CPCompressor;
public static double CompressionRatio;

// Gas Turbine Variables.
public static double CPTurbine;
public static double ExpansionRatio;

// Range Variables.
public static double WindTurbinePowerFrom;
public static double WindTurbinePowerTo;
public static double WindTurbinePowerStep;

public static double CompressorPowerFrom;
public static double CompressorPowerTo;
public static double CompressorPowerStep;

public static double AirStorageVolumeFrom;
public static double AirStorageVolumeTo;
public static double AirStorageVolumeStep;

#endregion User Input Values

#region Function Values

public static double DOD;
public static double MAL;
public static double CompressedAirDensity;
public static double CombustionChamberTemperature;
public static double GPower;

#endregion Function Values

#region Excel Imported Lists

// Import Excel.
public static List<double> WindSpeedList = new List<double>();
public static List<double> EnergyDemandList = new List<double>();
public static List<double> CelciusList = new List<double>();

// Default Excel.
public static List<double> EnergyDemandPerFamily = new List<double>();
public static List<double> DefaultCelciusList = new List<double>();
```

```
    public static List<double> EnergyConsumptionList = new List<double>();

    #endregion Excel Imported Lists

    #endregion Variables

    #region Animation Function

    private void pageFadeOut(int currentPage)
    {
      switch (currentPage)
      {
        case 1:
          Storyboard StartUnload = (Storyboard)FindResource("StartOut");
          StartUnload.Begin(this);
          break;
        case 2:
          Storyboard Step1Unload = (Storyboard)FindResource("Step1Out");
          Step1Unload.Begin(this);
          break;
        case 3:
          Storyboard Step2Unload = (Storyboard)FindResource("Step2Out");
          Step2Unload.Begin(this);
          break;
        case 4:
          Storyboard Step3Unload = (Storyboard)FindResource("Step3Out");
          Step3Unload.Begin(this);
          break;
        case 5:
          Storyboard FinishUnload = (Storyboard)FindResource("FinishOut");
          FinishUnload.Begin(this);
          break;
        default:
          break;
      }
    }

    #endregion Animation Function

    #region Start Window Code

    private void StartStep1btn_Click(object sender, RoutedEventArgs e)
    {
      pageFadeOut(currentPage);
      Storyboard Step1Load = (Storyboard)FindResource("Step1In");
```

```
        Step1Load.Begin(this);
        currentPage = 2;
    }

    #endregion Start Window Code

    #region Step1 Window Code

    private void StartStep2btn_Click(object sender,
System.Windows.RoutedEventArgs e)
    {
        // Read From Specified Excel.
        Excel.Application xlApp;
        Excel.Workbook xlWorkBook;
        Excel.Worksheet xlWorkSheet;
        Excel.Range range;
        xlApp = new Excel.Application();
        xlWorkBook = xlApp.Workbooks.Open(filename, 0, true, 5, "", "", true,
Microsoft.Office.Interop.Excel.XlPlatform.xlWindows, "\t", false, false, 0, true, 1, 0);
        xlWorkSheet = xlWorkBook.Worksheets.get_Item(1);
        range = xlWorkSheet.UsedRange;

        // Read from Template Excel.
        Excel.Application xlApp2;
        Excel.Workbook xlWorkBook2;
        Excel.Worksheet xlWorkSheet2;
        Excel.Range range2;
        xlApp2 = new Excel.Application();
        xlWorkBook2 = xlApp2.Workbooks.Open(defaultExcelFileName, 0, true, 5, "",
"", true, Microsoft.Office.Interop.Excel.XlPlatform.xlWindows, "\t", false, false, 0,
true, 1, 0);
        xlWorkSheet2 = xlWorkBook2.Worksheets.get_Item(1);
        range2 = xlWorkSheet2.UsedRange;

        bool valuesCorrectStep1 = true;

        if (DefaultEnergyDemandCheckBox.IsChecked == true)
        {
            if (Int32.TryParse(ResidentsTxtBox.Text, out Residents))
            {
                MessageBox.Show("Residents are not a valid number. Please give a
correct value.");
                ResidentsTxtBox.Text = "";
                valuesCorrectStep1 = false;
            }
            if (Residents <= 0)
```

```
            {
                MessageBox.Show("Residents must be above zero. Please give a correct
value.");
                ResidentsTxtBox.Text = "";
                valuesCorrectStep1 = false;
            }

            if (DefaultTemperatureListCheckBox.IsChecked == true)
            {
                // Read WindSpeedList from excel specified.
                // Read EnergyDemandPerFamily from default excel.
                // Load CelciusList from predefined excel.
                for (int i = 0; i < 8784; i++)
                {
                    WindSpeedList.Add((range.Cells[i + 2, 1] as Excel.Range).Value2);
                    EnergyDemandPerFamily.Add((range2.Cells[i + 2, 1] as
Excel.Range).Value2);
                    CelciusList.Add((range2.Cells[i + 2, 2] as Excel.Range).Value2);
                }

                EnergyConsumptionList =
CAESFunctions.EnergyConsumptionList(EnergyDemandPerFamily, Residents);
            }
            else
            {
                // Read WindSpeedList from excel specified.
                // Read EnergyDemandPerFamily from default excel.
                // Load CelciusList from predefined excel.
                for (int i = 0; i < 8784; i++)
                {
                    WindSpeedList.Add((range.Cells[i + 2, 1] as Excel.Range).Value2);
                    EnergyDemandPerFamily.Add((range2.Cells[i + 2, 1] as
Excel.Range).Value2);
                    CelciusList.Add((range.Cells[i + 2, 3] as Excel.Range).Value2);
                }

                EnergyConsumptionList =
CAESFunctions.EnergyConsumptionList(EnergyDemandPerFamily, Residents);
            }
        }
        else
        {
            if (DefaultTemperatureListCheckBox.IsChecked == true)
            {
                // Read WindSpeedList from excel specified.
                // Read EnergyDemandList from excel given.
```

```
            // Load CelciusList from predefined excel

            for (int i = 0; i < 8784; i++)
            {
              WindSpeedList.Add((range.Cells[i + 2, 1] as Excel.Range).Value2);
              EnergyDemandList.Add((range.Cells[i + 2, 2] as Excel.Range).Value2);
              CelciusList.Add((range2.Cells[i + 2, 2] as Excel.Range).Value2);
            }

            EnergyConsumptionList = EnergyDemandList;
          }
          else
          {
            // Read WindSpeedList from excel specified.
            // Read EnergyDemandList from excel given.
            // Read from 3rd Column the Celcius list.

            for (int i = 0; i < 8784; i++)
            {
              WindSpeedList.Add((range.Cells[i + 2, 1] as Excel.Range).Value2);
              EnergyDemandList.Add((range.Cells[i + 2, 2] as Excel.Range).Value2);
              CelciusList.Add((range.Cells[i + 2, 3] as Excel.Range).Value2);
            }

            EnergyConsumptionList = EnergyDemandList;
          }
        }

        if (valuesCorrectStep1)
        {
          // Animation to Step 2.
          pageFadeOut(currentPage);
          Storyboard Step2Load = (Storyboard)FindResource("Step2In");
          Step2Load.Begin(this);
          currentPage = 3;
        }
      }

      private void CheckBox_Checked(object sender,
System.Windows.RoutedEventArgs e)
      {
        ResidentsTxtBox.Visibility = Residentslbl.Visibility = Visibility.Visible;
      }

      private void CheckBox_Unchecked(object sender,
System.Windows.RoutedEventArgs e)
```

```
    {
      ResidentsTxtBox.Visibility = Residentslbl.Visibility = Visibility.Collapsed;
    }

    private void BrowseExcelFileBtn_Click(object sender, RoutedEventArgs e)
    {
      // Configure open file dialog box
      Microsoft.Win32.OpenFileDialog dlg = new Microsoft.Win32.OpenFileDialog();
      dlg.DefaultExt = ".xls"; // Default file extension
      dlg.Filter = "Excel Files (.xls)|*.xls"; // Filter files by extension
      dlg.Title = "Select the Excel data file";

      // Show open file dialog box
      Nullable<bool> result = dlg.ShowDialog();

      // Process open file dialog box results
      if (result == true)
      {
        // Open document
        filename = dlg.FileName;
        ExcelFilePathTxtBox.Text = dlg.FileName;
      }
    }

    private void releaseObject(object obj)
    {
      try
      {
        System.Runtime.InteropServices.Marshal.ReleaseComObject(obj);
        obj = null;
      }
      catch (Exception ex)
      {
        obj = null;
        MessageBox.Show("Unable to release the Object " + ex.ToString());
      }
      finally
      {
        GC.Collect();
      }
    }

#endregion Step1 Window Code

#region Step2 Window Code
```

```
        private void GoToStep1btn_Click(object sender,
System.Windows.RoutedEventArgs e)
    {
      // Animation back to Step 1.
      pageFadeOut(currentPage);
      Storyboard Step1Load = (Storyboard)FindResource("Step1In");
      Step1Load.Begin(this);
      currentPage = 2;
    }

    private void StartStep3btn_Click(object sender,
System.Windows.RoutedEventArgs e)
    {
      // Get Textbox variables.

      bool valuesCorrect = true;

      #region HU

      if (!Double.TryParse(HUTxtBox.Text, out HU))
      {
        MessageBox.Show("HU does not have a valid number. Please give a correct
value.");
        HUTxtBox.Text = "";
        valuesCorrect = false;
      }
      if (HU <= 0)
      {
        MessageBox.Show("HU cannot have a negative or zero value. Please give a
correct value.");
        HUTxtBox.Text = "";
        valuesCorrect = false;
      }

      #endregion HU

      #region Lamda

      if (!Double.TryParse(LamdaTxtBox.Text, out Lamda))
      {
        MessageBox.Show("Air Ratio does not have a valid number. Please give a
correct value.");
        LamdaTxtBox.Text = "";
        valuesCorrect = false;
      }
      if (Lamda <= 0)
```

```
        {
            MessageBox.Show("Air Ratio cannot have a negative or zero value. Please
give a correct value.");
            LamdaTxtBox.Text = "";
            valuesCorrect = false;
        }

        #endregion Lamda

        #region StorageTemperature

        if (!Double.TryParse(CavernStorageTempTxtBox.Text, out
StorageTemperature))
        {
            MessageBox.Show("Cavern Storage Temperature does not have a valid
number. Please give a correct value.");
            CavernStorageTempTxtBox.Text = "";
            valuesCorrect = false;
        }
        if (StorageTemperature <= 0)
        {
            MessageBox.Show("Cavern Storage Temperature cannot have a negative or
zero value. Please give a correct value.");
            CavernStorageTempTxtBox.Text = "";
            valuesCorrect = false;
        }

        #endregion StorageTemperature

        #region AirHeatCapacity

        if (!Double.TryParse(AirHeatCapacityTxtBox.Text, out AirHeatCapacity))
        {
            MessageBox.Show("Air Heat Capacity does not have a valid number. Please
give a correct value.");
            AirHeatCapacityTxtBox.Text = "";
            valuesCorrect = false;
        }
        if (AirHeatCapacity <= 0)
        {
            MessageBox.Show("Air Heat Capacity cannot have a negative or zero value.
Please give a correct value.");
            AirHeatCapacityTxtBox.Text = "";
            valuesCorrect = false;
        }
```

```
        #endregion AirHeatCapacity

        #region GasHeatCapacity

        if (!Double.TryParse(GasHeatCapacityTxtBox.Text, out GasHeatCapacity))
        {
          MessageBox.Show("Gas Heat Capacity does not have a valid number. Please
give a correct value.");
          GasHeatCapacityTxtBox.Text = "";
          valuesCorrect = false;
        }
        if (GasHeatCapacity <= 0)
        {
          MessageBox.Show("Gas Heat Capacity cannot have a negative or zero
value. Please give a correct value.");
          GasHeatCapacityTxtBox.Text = "";
          valuesCorrect = false;
        }

        #endregion GasHeatCapacity

        #region CPBrayton

        if (!Double.TryParse(CPBraytonTxtBox.Text, out CPBrayton))
        {
          MessageBox.Show("Brayton Cycle Efficiency does not have a valid number.
Please give a correct value.");
          CPBraytonTxtBox.Text = "";
          valuesCorrect = false;
        }
        if ((CPBrayton <= 0)||(CPBrayton > 1))
        {
          MessageBox.Show("Brayton Cycle Efficiency takes values in the range (0,1] .
Please give a correct value.");
          CPBraytonTxtBox.Text = "";
          valuesCorrect = false;
        }

        #endregion CPBrayton

        #region CPTurbine

        if (!Double.TryParse(GasTurbineEfficiencyTxtBox.Text, out CPTurbine))
        {
          MessageBox.Show("Gas Turbine Efficiency does not have a valid number.
Please give a correct value.");
```

```
            GasTurbineEfficiencyTxtBox.Text = "";
            valuesCorrect = false;
        }
        if ((CPTurbine <= 0) || (CPTurbine > 1))
        {
            MessageBox.Show("Gas Turbine Efficiency takes values in the range (0,1] .
Please give a correct value.");
            GasTurbineEfficiencyTxtBox.Text = "";
            valuesCorrect = false;
        }

        #endregion CPTurbine

        #region ExpansionRatio

        if (!Double.TryParse(GasTurbinePressureRatioTxtBox.Text, out
ExpansionRatio))
        {
            MessageBox.Show("Expansion Ratio does not have a valid number. Please
give a correct value.");
            GasTurbinePressureRatioTxtBox.Text = "";
            valuesCorrect = false;
        }
        if (ExpansionRatio <= 0)
        {
            MessageBox.Show("Expansion Ratio cannot have a negative or zero value.
Please give a correct value.");
            GasTurbinePressureRatioTxtBox.Text = "";
            valuesCorrect = false;
        }

        #endregion ExpansionRatio

        #region CPCompressor

        if (!Double.TryParse(CompressorEfficiencyTxtBox.Text, out CPCompressor))
        {
            MessageBox.Show("Compressor Efficiency does not have a valid number.
Please give a correct value.");
            CompressorEfficiencyTxtBox.Text = "";
            valuesCorrect = false;
        }
        if ((CPCompressor <= 0) || (CPCompressor > 1))
        {
            MessageBox.Show("Compressor Efficiency takes values in the range (0,1] .
Please give a correct value.");
```

```
            CompressorEfficiencyTxtBox.Text = "";
            valuesCorrect = false;
        }

        #endregion CPCompressor

        #region CompressionRatio

        if (!Double.TryParse(CompressorPressureRatioTxtBox.Text, out
CompressionRatio))
        {
            MessageBox.Show("Compression Ratio does not have a valid number.
Please give a correct value.");
            CompressorPressureRatioTxtBox.Text = "";
            valuesCorrect = false;
        }
        if (CompressionRatio <= 0)
        {
            MessageBox.Show("Compression Ratio cannot have a negative or zero
value. Please give a correct value.");
            CompressorPressureRatioTxtBox.Text = "";
            valuesCorrect = false;
        }

        #endregion CompressionRatio

        if (valuesCorrect == true)
        {
            MAL = CAESFunctions.MAL(StoichiometricRatio, Lamda);
            CombustionChamberTemperature =
CAESFunctions.CombustionChamberTemperature(AirHeatCapacity,
StorageTemperature, MAL, HU, GasHeatCapacity);

            if (CombustionChamberTemperature > 1700)
            {
                MessageBox.Show("Combustion Chamber Temperature higher than
1700K!/nAdjust one of the following:/n-HU/n-Air Ratio/n-Storage Temperature/n-Air
Heat Capacity/n-Gas Heat Capacity");
            }
            else
            {
                // Calculate hidden variables that are range variable independent.
                DOD = CAESFunctions.DOD(CompressionRatio, ExpansionRatio);
                CompressedAirDensity =
CAESFunctions.CompressedAirDensity(CompressionRatio, StorageTemperature);
                GPower = CAESFunctions.GPower(AdiabaticCoefficient);
```

```
            // Animation to Step 3.
            pageFadeOut(currentPage);
            Storyboard Step3Load = (Storyboard)FindResource("Step3In");
            Step3Load.Begin(this);
            currentPage = 4;
          }
        }
    }

    #endregion Step2 Window Code

    #region Step3 Window Code

                private void GoToStep2btn_Click(object sender,
System.Windows.RoutedEventArgs e)
    {
       // Animation back to Step 2.
       pageFadeOut(currentPage);
       Storyboard Step2Load = (Storyboard)FindResource("Step2In");
       Step2Load.Begin(this);
       currentPage = 3;
    }

    private void Exportbtn_Click(object sender, System.Windows.RoutedEventArgs
e)
    {
       // Get Textbox variables.

       bool valuesCorrectStep3 = true;

       #region WindTurbinePower

       #region WindTurbinePowerFrom

       if (!Double.TryParse(WindTurbinePowerFromTxtBox.Text, out
WindTurbinePowerFrom))
       {
          MessageBox.Show("Wind Turbine Power From Text does not have a valid
number. Please give a correct value.");
          WindTurbinePowerFromTxtBox.Text = "";
          valuesCorrectStep3 = false;
       }
       if (WindTurbinePowerFrom < 0)
       {
```

```
            MessageBox.Show("Wind Turbine Power From Text Box cannot have a
negative value. Please give a correct value.");
            WindTurbinePowerFromTxtBox.Text = "";
            valuesCorrectStep3 = false;
        }

        #endregion WindTurbinePowerFrom

        #region WindTurbinePowerTo

        if (!Double.TryParse(WindTurbinePowerToTxtBox.Text, out
WindTurbinePowerTo))
        {
            MessageBox.Show("Wind Turbine Power To Text does not have a valid
number. Please give a correct value.");
            WindTurbinePowerToTxtBox.Text = "";
            valuesCorrectStep3 = false;
        }
        if (WindTurbinePowerTo < 0)
        {
            MessageBox.Show("Wind Turbine Power To Text Box cannot have a
negative value. Please give a correct value.");
            WindTurbinePowerToTxtBox.Text = "";
            valuesCorrectStep3 = false;
        }

        #endregion WindTurbinePowerTo

        #region WindTurbinePowerStep

        if (!Double.TryParse(WindTurbinePowerStepTxtBox.Text, out
WindTurbinePowerStep))
        {
            MessageBox.Show("Wind Turbine Power Step Text does not have a valid
number. Please give a correct value.");
            WindTurbinePowerStepTxtBox.Text = "";
            valuesCorrectStep3 = false;
        }
        if (WindTurbinePowerStep <= 0)
        {
            MessageBox.Show("Wind Turbine Power Step Text Box must have a positive
value. Please give a correct value.");
            WindTurbinePowerStepTxtBox.Text = "";
            valuesCorrectStep3 = false;
        }
```

```
        #endregion WindTurbinePower

        #region CheckFromToValues

        if (WindTurbinePowerFrom > WindTurbinePowerTo)
        {
            MessageBox.Show("Wind Turbine Power To Text Box must be greater or
equal to Wind Turbine Power From Text Box. Please give a correct value.");
            valuesCorrectStep3 = false;
        }

        #endregion CheckFromToValues

        #endregion WindTurbinePower

        #region AirCompressorPower

        #region CompressorPowerFrom

        if (!Double.TryParse(AirCompressorFromTxtBox.Text, out
CompressorPowerFrom))
        {
            MessageBox.Show("Air Compressor Power From Text does not have a valid
number. Please give a correct value.");
            AirCompressorFromTxtBox.Text = "";
            valuesCorrectStep3 = false;
        }
        if (CompressorPowerFrom < 0)
        {
            MessageBox.Show("Air Compressor Power From Text Box cannot have a
negative value. Please give a correct value.");
            AirCompressorFromTxtBox.Text = "";
            valuesCorrectStep3 = false;
        }

        #endregion CompressorPowerFrom

        #region CompressorPowerTo

        if (!Double.TryParse(AirCompressorToTxtBox.Text, out CompressorPowerTo))
        {
            MessageBox.Show("Air Compressor Power To Text does not have a valid
number. Please give a correct value.");
            AirCompressorToTxtBox.Text = "";
            valuesCorrectStep3 = false;
        }
```

```
        if (CompressorPowerTo < 0)
        {
          MessageBox.Show("Air Compressor Power To Text Box cannot have a
negative value. Please give a correct value.");
          AirCompressorToTxtBox.Text = "";
          valuesCorrectStep3 = false;
        }

        #endregion CompressorPowerTo

        #region CompressorPowerStep

        if (!Double.TryParse(AirCompressorStepTxtBox.Text, out
CompressorPowerStep))
        {
          MessageBox.Show("Air Compressor Power Step Text does not have a valid
number. Please give a correct value.");
          AirCompressorStepTxtBox.Text = "";
          valuesCorrectStep3 = false;
        }
        if (CompressorPowerStep <= 0)
        {
          MessageBox.Show("Air Compressor Power Step Text Box must have a
positive value. Please give a correct value.");
          AirCompressorStepTxtBox.Text = "";
          valuesCorrectStep3 = false;
        }

        #endregion CompressorPowerStep

        #region CheckFromToValues

        if (CompressorPowerFrom > CompressorPowerTo)
        {
          MessageBox.Show("Air Compressor Power To Text Box must be greater or
equal to Air Compressor Power From Text Box. Please give a correct value.");
          valuesCorrectStep3 = false;
        }

        #endregion CheckFromToValues

        #endregion AirCompressorPower

        #region AirStorageVolume

        #region AirStorageVolumeFrom
```

```
        if (!Double.TryParse(AirStorageFromTxtBox.Text, out AirStorageVolumeFrom))
        {
            MessageBox.Show("Air Storage Volume From Text does not have a valid
number. Please give a correct value.");
            AirStorageFromTxtBox.Text = "";
            valuesCorrectStep3 = false;
        }
        if (AirStorageVolumeFrom < 0)
        {
            MessageBox.Show("Air Storage Volume From Text Box cannot have a
negative value. Please give a correct value.");
            AirStorageFromTxtBox.Text = "";
            valuesCorrectStep3 = false;
        }

        #endregion AirStorageVolumeFrom

        #region AirStorageVolumeTo

        if (!Double.TryParse(AirStorageToTxtBox.Text, out AirStorageVolumeTo))
        {
            MessageBox.Show("Air Storage Volume To Text does not have a valid
number. Please give a correct value.");
            AirStorageToTxtBox.Text = "";
            valuesCorrectStep3 = false;
        }
        if (AirStorageVolumeTo < 0)
        {
            MessageBox.Show("Air Storage Volume To Text Box cannot have a negative
value. Please give a correct value.");
            AirStorageToTxtBox.Text = "";
            valuesCorrectStep3 = false;
        }

        #endregion AirStorageVolumeTo

        #region AirStorageVolumeStep

        if (!Double.TryParse(AirStorageStepTxtBox.Text, out AirStorageVolumeStep))
        {
            MessageBox.Show("Air Storage Volume Step Text does not have a valid
number. Please give a correct value.");
            AirStorageStepTxtBox.Text = "";
            valuesCorrectStep3 = false;
        }
```

```
        if (CompressorPowerStep <= 0)
        {
          MessageBox.Show("Air Storage Volume Step Text Box must have a positive
value. Please give a correct value.");
          AirStorageStepTxtBox.Text = "";
          valuesCorrectStep3 = false;
        }

        #endregion AirStorageVolumeStep

        #region CheckFromToValues

        if (AirStorageVolumeFrom > AirStorageVolumeTo)
        {
          MessageBox.Show("Air Storage Volume To Text Box must be greater or
equal to Air Storage Volume From Text Box. Please give a correct value.");
          valuesCorrectStep3 = false;
        }

        #endregion CheckFromToValues

        #endregion AirStorageVolume

        if (valuesCorrectStep3)
        {
          // Construct Lists.
          List<double> KelvinList = new List<double>();
          List<double> PowerProductionList = new List<double>();
          List<double> EnergyNeedList = new List<double>();
          List<double> EnergyRejectionList = new List<double>();
          List<double> PowerStorageList = new List<double>();
          List<double> MassAirCompressorList = new List<double>();
          List<double> CAESMassGasList = new List<double>();
          List<double> CAESMassAirList = new List<double>();
          List<double> CAESMassFuelList = new List<double>();
          List<double> AirStorageLevelList = new List<double>();
          List<double> AirStorageDifferenceList = new List<double>();
          List<double> MAActualList = new List<double>();
          List<double> DualTurbinePowerList = new List<double>();
          List<double> FinalEnergyNeedList = new List<double>();

          // Sum Values.
          double FinalEnergyNeedSum;
          double CAESNonCoverageHours;
          double AnnualCAESMassFuel;
          double AnnualDualMassFuel;
```

```
double AnnualTotalMassFuel;
double AnnualBraytonMassFuel;
double WindPowerExcess;
double HeatRate;
double DMF;
double WindCalmsTotalCounter;
double MaxConcecutiveWindCalms;

// Calculate list variables that are range variable independent.
KelvinList = CAESFunctions.KelvinList(CelciusList);

double windTurbinePower = WindTurbinePowerFrom;
double CompressorPower = CompressorPowerFrom;
double airStorageVolume = AirStorageVolumeFrom;

// Open excel file to write with a save to file Dialog.
Excel.Application xlApp;
Excel.Workbook xlWorkBook;
Excel.Worksheet xlWorkSheet;
object misValue = System.Reflection.Missing.Value;

xlApp = new Excel.Application();
xlWorkBook = xlApp.Workbooks.Add(misValue);

xlWorkSheet = xlWorkBook.Worksheets.get_Item(1);

// Write variable names to excel file.
xlWorkSheet.Cells[1, 2] = "Wind Turbine Power (MW)";
xlWorkSheet.Cells[1, 3] = "Compressor Power (MW)";
xlWorkSheet.Cells[1, 4] = "Air Storage Volume (m^3)";
xlWorkSheet.Cells[1, 5] = "CAES Energy Deficit (MWh/year)";
xlWorkSheet.Cells[1, 6] = "Wind Power Excess (MWh/year)";
xlWorkSheet.Cells[1, 7] = "CAES Non-coverage (h)";
xlWorkSheet.Cells[1, 8] = "CAES Fuel Mass (ton/year)";
xlWorkSheet.Cells[1, 9] = "Dual Mode Fuel Mass (ton/year)";
xlWorkSheet.Cells[1, 10] = "Total Fuel Mass (ton/year)";
xlWorkSheet.Cells[1, 11] = "Brayton cycle Fuel Mass (ton/year)";
xlWorkSheet.Cells[1, 12] = "Heat Rate (kWhng/kWhe)";
xlWorkSheet.Cells[1, 13] = "Fuel Economy (ton/year)";
xlWorkSheet.Cells[1, 14] = "Total Wind Calms";
xlWorkSheet.Cells[1, 15] = "Consecutive Wind Calms";

xlWorkSheet.Cells[2, 1] = "Compressor Efficiency";
xlWorkSheet.Cells[4, 1] = "Compression Ratio";
xlWorkSheet.Cells[6, 1] = "Gas Turbine Efficiency";
xlWorkSheet.Cells[8, 1] = "Expansion Ratio";
```

```
            xlWorkSheet.Cells[10, 1] = "Brayton Cycle Efficiency";
            xlWorkSheet.Cells[12, 1] = "Storage Temperature (K)";
            xlWorkSheet.Cells[14, 1] = "Air Heat Capacity (KJ/Kg*K)";
            xlWorkSheet.Cells[16, 1] = "Gas Heat Capacity (KJ/Kg*K)";
            xlWorkSheet.Cells[18, 1] = "Fuel Calorific value (MJ/Kg)";
            xlWorkSheet.Cells[20, 1] = "Air Ratio (λ)";
            xlWorkSheet.Cells[22, 1] = "Adiabatic Coefficient";
            xlWorkSheet.Cells[24, 1] = "Stoichiometric Ratio";

            // Write variables first to excel.
            xlWorkSheet.Cells[3, 1] = CPCompressor.ToString();
            xlWorkSheet.Cells[5, 1] = CPCompressor.ToString();
            xlWorkSheet.Cells[7, 1] = CompressionRatio.ToString();
            xlWorkSheet.Cells[9, 1] = CPTurbine.ToString();
            xlWorkSheet.Cells[11, 1] = ExpansionRatio.ToString();
            xlWorkSheet.Cells[13, 1] = CPBrayton.ToString();
            xlWorkSheet.Cells[15, 1] = StorageTemperature.ToString();
            xlWorkSheet.Cells[17, 1] = AirHeatCapacity.ToString();
            xlWorkSheet.Cells[19, 1] = HU.ToString();
            xlWorkSheet.Cells[21, 1] = Lamda.ToString();
            xlWorkSheet.Cells[23, 1] = AdiabaticCoefficient.ToString();
            xlWorkSheet.Cells[25, 1] = StoichiometricRatio.ToString();

            double counter = 2;

            while (windTurbinePower <= WindTurbinePowerTo)
            {
               while (CompressorPower <= CompressorPowerTo)
               {
                  while (airStorageVolume <= AirStorageVolumeTo)
                  {
                     // Calculate Compressor Max Air.
                     double CompressorMaxAir =
CAESFunctions.CompressorMaxAir(CompressedAirDensity, airStorageVolume);

                     // Fill Lists.
                     PowerProductionList =
CAESFunctions.PowerProductionList(WindSpeedList, windTurbinePower);
                     EnergyNeedList =
CAESFunctions.EnergyNeedList(EnergyConsumptionList, PowerProductionList);
                     EnergyRejectionList =
CAESFunctions.EnergyRejectionList(EnergyConsumptionList, PowerProductionList);
                     PowerStorageList =
CAESFunctions.PowerStorageList(EnergyNeedList, CompressorPower,
CPCompressor);
```

```
                MassAirCompressorList =
CAESFunctions.MassAirCompressorList(PowerStorageList, KelvinList,
AirHeatCapacity, CompressionRatio, GPower);
                CAESMassGasList = CAESFunctions.CAESMassGasList(EnergyNeedList,
GasHeatCapacity, CombustionChamberTemperature, ExpansionRatio, GPower,
CPTurbine);
                CAESMassAirList = CAESFunctions.CAESMassAirList(CAESMassGasList,
GasHeatCapacity, CombustionChamberTemperature, AirHeatCapacity, HU, MAL,
StorageTemperature);
                CAESMassFuelList =
CAESFunctions.CAESMassFuelList(CAESMassAirList, CAESMassGasList);
                AirStorageLevelList =
CAESFunctions.AirStorageLevelList(EnergyNeedList, MassAirCompressorList,
CAESMassAirList, CompressorMaxAir, DOD);
                AirStorageDifferenceList =
CAESFunctions.AirStorageDifferenceList(AirStorageLevelList, CompressorMaxAir,
DOD);
                MAActualList =
CAESFunctions.MAActualList(AirStorageDifferenceList);
                DualTurbinePowerList =
CAESFunctions.DualTurbinePowerList(MAActualList, MAL,
CombustionChamberTemperature, GasHeatCapacity, ExpansionRatio, GPower,
CPTurbine);
                FinalEnergyNeedList =
CAESFunctions.FinalEnergyNeedList(PowerProductionList, EnergyConsumptionList,
DualTurbinePowerList);

                // Calculate Sum Values.
                FinalEnergyNeedSum =
CAESFunctions.FinalEnergyNeedSum(FinalEnergyNeedList);
                CAESNonCoverageHours =
CAESFunctions.CAESNonCoverageHours(FinalEnergyNeedList);
                AnnualCAESMassFuel =
CAESFunctions.AnnualCAESMassFuel(MAActualList, MAL);
                AnnualDualMassFuel =
CAESFunctions.AnnualDualMassFuel(FinalEnergyNeedSum, HU, CPBrayton);
                AnnualTotalMassFuel =
CAESFunctions.AnnualTotalMassFuel(AnnualCAESMassFuel, AnnualDualMassFuel);
                AnnualBraytonMassFuel =
CAESFunctions.AnnualBraytonMassFuel(EnergyNeedList, HU, CPBrayton);
                WindPowerExcess =
CAESFunctions.WindPowerExcess(EnergyRejectionList);
                HeatRate = CAESFunctions.HeatRate(DualTurbinePowerList,
AnnualCAESMassFuel, HU);
                DMF = CAESFunctions.DMF(AnnualBraytonMassFuel,
AnnualTotalMassFuel);
```

```
                WindCalmsTotalCounter =
CAESFunctions.WindCalms(WindSpeedList).WindCalmsTotalCounter;
                MaxConcecutiveWindCalms =
CAESFunctions.WindCalms(WindSpeedList).MaxConcecutiveWindCalms;

                // Export Sum Values to excel
                xlWorkSheet.Cells[counter, 2] =
windTurbinePower.ToString(CultureInfo.InvariantCulture);
                xlWorkSheet.Cells[counter, 3] =
CompressorPower.ToString(CultureInfo.InvariantCulture);
                xlWorkSheet.Cells[counter, 4] =
airStorageVolume.ToString(CultureInfo.InvariantCulture);
                xlWorkSheet.Cells[counter, 5] =
FinalEnergyNeedSum.ToString(CultureInfo.InvariantCulture);
                xlWorkSheet.Cells[counter, 6] =
WindPowerExcess.ToString(CultureInfo.InvariantCulture);
                xlWorkSheet.Cells[counter, 7] =
CAESNonCoverageHours.ToString(CultureInfo.InvariantCulture);
                xlWorkSheet.Cells[counter, 8] =
AnnualCAESMassFuel.ToString(CultureInfo.InvariantCulture);
                xlWorkSheet.Cells[counter, 9] =
AnnualDualMassFuel.ToString(CultureInfo.InvariantCulture);
                xlWorkSheet.Cells[counter, 10] =
AnnualTotalMassFuel.ToString(CultureInfo.InvariantCulture);
                xlWorkSheet.Cells[counter, 11] =
AnnualBraytonMassFuel.ToString(CultureInfo.InvariantCulture);
                xlWorkSheet.Cells[counter, 12] =
HeatRate.ToString(CultureInfo.InvariantCulture);
                xlWorkSheet.Cells[counter, 13] =
DMF.ToString(CultureInfo.InvariantCulture);
                xlWorkSheet.Cells[counter, 14] =
WindCalmsTotalCounter.ToString(CultureInfo.InvariantCulture);
                xlWorkSheet.Cells[counter, 15] =
MaxConcecutiveWindCalms.ToString(CultureInfo.InvariantCulture);

                counter++;

                airStorageVolume += AirStorageVolumeStep;
            }

            airStorageVolume = AirStorageVolumeFrom;
            CompressorPower += CompressorPowerStep;
        }

        CompressorPower = CompressorPowerFrom;
        windTurbinePower += WindTurbinePowerStep;
```

```
            }

            // Configure save file dialog box
            Microsoft.Win32.SaveFileDialog dlg2 = new
  Microsoft.Win32.SaveFileDialog();
            dlg2.FileName = "Results"; // Default file name
            dlg2.DefaultExt = ".xls"; // Default file extension
            dlg2.Filter = "Excel Files (.xls)|*.xls"; // Filter files by extension
            dlg2.Title = "Save Excel data file";

            // Show save file dialog box
            Nullable<bool> result = dlg2.ShowDialog();

            // Process save file dialog box results
            if (result == true)
            {
              // Save document
              string savefilename = dlg2.FileName;

              xlWorkBook.SaveAs(savefilename,
  Excel.XlFileFormat.xlWorkbookNormal, misValue, misValue, misValue, misValue,
  Excel.XlSaveAsAccessMode.xlExclusive, misValue, misValue, misValue, misValue,
  misValue);
              xlWorkBook.Close(true, misValue, misValue);
              xlApp.Quit();

              releaseObject(xlWorkSheet);
              releaseObject(xlWorkBook);
              releaseObject(xlApp);
          }

          // Animation to Finish.
          pageFadeOut(currentPage);
          Storyboard FinishLoad = (Storyboard)FindResource("FinishIn");
          FinishLoad.Begin(this);
          currentPage = 5;
      }
    }

    #endregion Step3 Window Code

    #region Finish Window Code

    private void StartNewStudyBtn_Click(object sender,
  System.Windows.RoutedEventArgs e)
    {
```

```
        // Ask if the user wants to reset or keep results.
        MessageBoxResult result = MessageBox.Show("Do you want to keep the
previous values?", "Keep previous values?", MessageBoxButton.YesNo,
MessageBoxImage.Question);

        switch (result)
        {
          case MessageBoxResult.Yes:

            break;
          case MessageBoxResult.No:

            #region Clear values on input boxes

            // Step 1.
            ExcelFilePathTxtBox.Clear();
            DefaultEnergyDemandCheckBox.IsChecked = false;
            ResidentsTxtBox.Clear();

            // Step 2.
            CompressorEfficiencyTxtBox.Clear();
            CompressorPressureRatioTxtBox.Clear();
            GasTurbineEfficiencyTxtBox.Clear();
            GasTurbinePressureRatioTxtBox.Clear();
            HUTxtBox.Clear();
            CavernStorageTempTxtBox.Clear();

            // Step 3.
            WindTurbinePowerFromTxtBox.Clear();
            WindTurbinePowerStepTxtBox.Clear();
            WindTurbinePowerToTxtBox.Clear();
            AirCompressorFromTxtBox.Clear();
            AirCompressorToTxtBox.Clear();
            AirCompressorStepTxtBox.Clear();
            AirStorageFromTxtBox.Clear();
            AirStorageStepTxtBox.Clear();
            AirStorageToTxtBox.Clear();

            #endregion Clear values on input boxes

            break;
        }

        // Animation to Step 1.
        pageFadeOut(currentPage);
        Storyboard Step1Load = (Storyboard)FindResource("Step1In");
```

```
            Step1Load.Begin(this);
            currentPage = 2;
        }

        #endregion Finish Window Code
    }
}
```

7.2.3 Αλγόριθμος εξισώσεων και υπολογισμών

```
using System;
using System.Collections.Generic;
using System.Linq;

namespace CAES
{
  class CAESFunctions
  {
    #region Function Values

    #region DOD

    /// <summary>
    /// Returns the DOD based on the Compressor Ratio and the Expansion Ratio.
    /// </summary>
    /// <param name="CompressionRatio">The Compression Ratio.</param>
    /// <param name="ExpansionRatio">The Expansion Ratio.</param>
    /// <returns></returns>
    public static double DOD(double CompressorRatio, double ExpansionRatio)
    {
      double DOD = (CompressorRatio - ExpansionRatio) / CompressorRatio;
      return DOD;
    }

    #endregion DOD

    #region MAL

    /// <summary>
    /// Returns the MAL based on the Stoichiometric Ratio and the Lamda.
    /// </summary>
    /// <param name="StoichiometricRatio"> The Stoichiometric Ratio.</param>
    /// <param name="Lamda">The Lamda.</param>
    /// <returns></returns>
    public static double MAL(double StoichiometricRatio, double Lamda)
```

```
        {
            double MAL = StoichiometricRatio * Lamda;
            return MAL;
        }

        #endregion MAL

        #region Compressed Air Density

        /// <summary>
        /// Returns the Compressed Air Density based on the Compressor Ratio and the
    Storage Temperature (Kg/m^3).
        /// </summary>
        /// <param name="CompressionRatio">The Compression Ratio.</param>
        /// <param name="StorageTemperature">The Storage Temperature.</param>
        /// <returns></returns>
        public static double CompressedAirDensity(double CompressionRatio, double
    StorageTemperature)
        {
            double CompressedAirDensity = CompressionRatio * 100000 / (287 *
    StorageTemperature);
            return CompressedAirDensity;
        }

        #endregion Compressed Air Density

        #region Combustion Chamber Temperature

        /// <summary>
        /// Returns the Combustion Chamber Temperature based on the Air Heat
    Capacity, the Storage Temperature, the MAL, the HU and the Gas Heat Capacity
    (Kelvin).
        /// </summary>
        /// <param name="AirHeatCapacity">The Air Heat Capacity
    (KJ/Kg*Kelvin).</param>
        /// <param name="StorageTemperature">The Storage Temperature
    (Kelvin).</param>
        /// <param name="MAL">The MAL.</param>
        /// <param name="HU">The HU.</param>
        /// <param name="GasHeatCapacity">The Gas Heat Capacity
    (KJ/Kg*Kelvin).</param>
        /// <returns></returns>
        public static double CombustionChamberTemperature (double AirHeatCapacity,
    double StorageTemperature, double MAL, double HU, double GasHeatCapacity)
        {
```

```
        double CombustionChamberTemperature = (AirHeatCapacity *
StorageTemperature * MAL + HU * 1000)/(GasHeatCapacity * (MAL+1));
        return CombustionChamberTemperature;
      }

      #endregion Combustion Chamber Temperature

      #region GPower

      /// <summary>
      /// Returns the G Power based on the Adiabatic Coefficient.
      /// </summary>
      /// <param name="AdiabaticCoefficient">The Adiabatic Coefficient.</param>
      /// <returns></returns>
      public static double GPower(double AdiabaticCoefficient)
      {
        double Gpower = (AdiabaticCoefficient - 1) / AdiabaticCoefficient;
        return Gpower;
      }

      #endregion GPower

      // Changes in every step.
      #region Compressor Max Air

      /// <summary>
      /// Returns the Compressor Max Air based on the Compressed Air Density and
the Air Storage Volume (Kg).
      /// </summary>
      /// <param name="CompressedAirDensity">The Compressed Air Density
(Kg/m^3).</param>
      /// <param name="AirStorageVolume">The Air Storage Volume (m^3).</param>
      /// <returns></returns>
      public static double CompressorMaxAir(double CompressedAirDensity, double
AirStorageVolume)
      {
        double CompressorMaxAir = CompressedAirDensity * AirStorageVolume;
        return CompressorMaxAir;
      }

      #endregion Compressor Max Air

    #endregion Function Values

    #region List Values
```

```
        #region Kelvin List

        /// <summary>
        /// Converts a List in Celcius to a List in Kelvin.
        /// </summary>
        /// <param name="CelciusList">Celcius List.</param>
        /// <returns></returns>
        public static List<double> KelvinList(List<double> CelciusList)
        {
          // Kelvin List constructor.
          List<double> KelvinList = new List<double>();

          double Kelvin;

          foreach (double Celcius in CelciusList)
          {
            Kelvin = 273.15 + Celcius;
            KelvinList.Add(Kelvin);
          }

          return KelvinList;
        }

        #endregion Kelvin List

        // All the others change in every step.

        #region Power Production List

        /// <summary>
        /// Returns a List of the Power Production of the Wind Turbine based on the
Wind Speed List and the Wind Turbine Power.
        /// </summary>
        /// <param name="WindSpeed">The Wind Speed List values (m/s).</param>
        /// <param name="WindTurbinePower">The Wind Turbine Power value
(MW).</param>
        /// <returns></returns>
        public static List<double> PowerProductionList(List<double> WindSpeedList,
double WindTurbinePower)
        {
          // PowerProductionList constructor.
          List<double> PowerProductionList = new List<double>();

          double PowerProduction;

          foreach (double WindSpeed in WindSpeedList)
```

```
            {
                if (WindSpeed < 4)
                {
                    PowerProduction = 0;
                }
                else if (WindSpeed < 8)
                {
                    PowerProduction = (0.0126 * (Math.Pow(WindSpeed, 2)) - 0.072 *
    WindSpeed + 0.1177) * WindTurbinePower;
                }
                else if (WindSpeed < 11.5)
                {
                    PowerProduction = (0.1546 * WindSpeed - 0.8811) * WindTurbinePower;
                }
                else if (WindSpeed < 14)
                {
                    PowerProduction = (-0.0206 * (Math.Pow(WindSpeed, 2)) + 0.5621 *
    WindSpeed - 2.8397) * WindTurbinePower;
                }
                else if (WindSpeed <= 25)
                {
                    PowerProduction = WindTurbinePower;
                }
                else // WindSpeed > 25
                {
                    PowerProduction = 0;
                }

                PowerProductionList.Add(PowerProduction);
            }

            return PowerProductionList;
        }

        #endregion Power Production List

        #region Energy Consumption List

        /// <summary>
        /// Returns a List of the Energy Consumption based on the Energy Demand per
    Family and the number of residents.
        /// </summary>
        /// <param name="EnergyDemandPerFamily">Energy Demand per Family
    List.</param>
        /// <param name="Residents">Number of Residents.</param>
        /// <returns></returns>
```

```
        public static List<double> EnergyConsumptionList(List<double>
    EnergyDemandPerFamily, int Residents)
        {
          // EnergyConsumptionList constructor.
          List<double> EnergyConsumptionList = new List<double>();

          double EnergyConsumption;

          foreach (double FamilyEnergyDemand in EnergyDemandPerFamily)
          {
            EnergyConsumption = (FamilyEnergyDemand * Residents ) / 4000000; //
    Note: 4 * 1.000.000
            EnergyConsumptionList.Add(EnergyConsumption);
          }

          return EnergyConsumptionList;
        }

        #endregion Energy Consumption List

        #region Energy Need List

        /// <summary>
        /// Returns a List of the Energy Need based on the Energy Consumption List and
     the Power Production List.
        /// </summary>
        /// <param name="EnergyConsumptionList">Energy Consumption List
    (MW).</param>
        /// <param name="PowerProductionList">Power Production List
    (MW).</param>
        /// <returns></returns>
        public static List<double> EnergyNeedList(List<double> EnergyConsumptionList,
    List<double> PowerProductionList)
        {
          // EnergyNeedList constructor.
          List<double> EnergyNeedList = new List<double>();

          double EnergyNeed;

          for (int i =0; i < EnergyConsumptionList.Count; i++)
          {
            EnergyNeed = PowerProductionList[i] - EnergyConsumptionList[i] ;
            EnergyNeedList.Add(EnergyNeed);
          }

          return EnergyNeedList;
```

```
        }

        #endregion Energy Need List

        #region Energy Rejection List

        /// <summary>
        /// Returns a List of the Energy Rejection based on the Energy Consumption List
and the Power Production List.
        /// </summary>
        /// <param name="EnergyConsumptionList">Energy Consumption List
(MW).</param>
        /// <param name="PowerProductionList">Power Production List
(MW).</param>
        /// <returns></returns>
        public static List<double> EnergyRejectionList(List<double>
EnergyConsumptionList, List<double> PowerProductionList)
        {
            // EnergyRejection constructor.
            List<double> EnergyRejectionList = new List<double>();

            double EnergyRejection;

            for (int i = 0; i < EnergyConsumptionList.Count; i++)
            {
                if (EnergyConsumptionList[i] < PowerProductionList[i])
                {
                    EnergyRejection = PowerProductionList[i] - EnergyConsumptionList[i];
                }
                else
                {
                    EnergyRejection = 0;
                }

                EnergyRejectionList.Add(EnergyRejection);
            }

            return EnergyRejectionList;
        }

        #endregion Energy Rejection List

        #region Power Storage List

        /// <summary>
```

```
        /// Returns the Power Storage List based on the Energy Need List, the
    Compressor Power and the CPCompressor (MW).
        /// </summary>
        /// <param name="EnergyNeedList">The Energy Need List (MW).</param>
        /// <param name="CompressorPower">The Compressor Power (MW).</param>
        /// <param name="CPCompressor">The CPCompressor.</param>
        /// <returns></returns>
        public static List<double> PowerStorageList(List<double> EnergyNeedList,
    double CompressorPower, double CPCompressor)
        {
            // PowerStorage List constructor.
            List<double> PowerStorageList = new List<double>();

            double PowerStorage;

            for (int i = 0; i < EnergyNeedList.Count; i++)
            {
              if(EnergyNeedList[i]>0)
              {
                if (EnergyNeedList[i] < CompressorPower)
                {
                  PowerStorage = EnergyNeedList[i] * CPCompressor;
                }
                else
                {
                  PowerStorage = CompressorPower * CPCompressor;
                }
             }
             else
             {
               PowerStorage=0;
             }

            PowerStorageList.Add(PowerStorage);
        }

        return PowerStorageList;
    }

    #endregion Power Storage List

    #region Mass Air Compressor List

    /// <summary>
    /// Returns the Mass Air Compressor List based on the Power Storage List, the
Kelvin List, the Air Heat Capacity, the Compression Ratio and the GPower (Kg/Hour).
```

```
    /// </summary>
    /// <param name="PowerStorageList">The Power Storage List (MW).</param>
    /// <param name="KelvinList">The Kelvin List (K).</param>
    /// <param name="AirHeatCapacity">The Air Heat Capacity
(KJ/(Kg*K)).</param>
    /// <param name="CompressionRatio">The Compression Ratio.</param>
    /// <param name="GPower">The GPower.</param>
    /// <returns></returns>
    public static List<double> MassAirCompressorList(List<double>
PowerStorageList, List<double> KelvinList, double AirHeatCapacity, double
CompressionRatio, double GPower)
    {
      // MassAirCompressor List constructor.
      List<double> MassAirCompressorList = new List<double>();

      double MassAirCompressor;

      for (int i = 0; i < PowerStorageList.Count; i++)
      {
        MassAirCompressor = 3600 * ((PowerStorageList[i] * 1000) / (KelvinList[i] *
AirHeatCapacity * (Math.Pow(CompressionRatio, GPower) - 1)));
        MassAirCompressorList.Add(MassAirCompressor);
      }

      return MassAirCompressorList;
    }

    #endregion Mass Air Compressor List

    #region CAES Mass Gas List

    /// <summary>
    /// Returns the CAES Mass Gas List based on the Energy Need List, the Gas Heat
Capacity, the Combustion Chamber Temperature, the Expansion Ratio, the CP
Turbine (Kg/sec).
    /// </summary>
    /// <param name="EnergyNeedList"> The Energy Need List (MW).</param>
    /// <param name="GasHeatCapacity">The Gas Heat Capacity
(KJ/(Kg*K)).</param>
    /// <param name="CombustionChamberTemperature">The Combustion
Chamber Temperature (K).</param>
    /// <param name="ExpansionRatio">The Expansion Ratio.</param>
    /// <param name="GPower">The GPower.</param>
    /// <param name="CPTurbine">The CP Turbine.</param>
    /// <returns></returns>
```

```
        public static List<double> CAESMassGasList(List<double> EnergyNeedList,
    double GasHeatCapacity, double CombustionChamberTemperature, double
    ExpansionRatio, double GPower, double CPTurbine)
        {
            // CAESMassGas List constructor.
            List<double> CAESMassGasList = new List<double>();

            double CAESMassGas;

            foreach (double EnergyNeed in EnergyNeedList)
            {
                if (EnergyNeed < 0)
                {
                    CAESMassGas = -EnergyNeed * 1000 / (GasHeatCapacity *
    CombustionChamberTemperature * (1 - (1 / (Math.Pow(ExpansionRatio, GPower))))
    * CPTurbine);
                    CAESMassGasList.Add(CAESMassGas);
                }
                else
                {
                    CAESMassGas = 0;
                    CAESMassGasList.Add(CAESMassGas);
                }
            }

            return CAESMassGasList;
        }

        #endregion CAES Mass Gas List

        #region CAES Mass Air List

        /// <summary>
        /// Returns the CAES Mass Air List based on the CAES Mass Gas List, the Gas
     Heat Capacity, the Combustion Chamber Temperature, the AirHeatCapacityList, the
     KelvinList, the HU, and the MAL (Kg/sec).
        /// </summary>
        /// <param name="CAESMassGasList"> The CAES Mass Gas List
    (Kg/sec).</param>
        /// <param name="GasHeatCapacity">The Gas Heat Capacity
    (KJ/(Kg*K)).</param>
        /// <param name="CombustionChamberTemperature">The Combustion
    Chamber Temperature (K).</param>
        /// <param name="AirHeatCapacity">The Air Heat Capacity
    (KJ/(Kg*K)).</param>
        /// <param name="HU">The HU of fuel (MJ/Kg).</param>
```

```
        /// <param name="MAL">The MA * L.</param>
        /// <param name="StorageTemperature">The Storage Temperature</param>
        /// <returns></returns>
        public static List<double> CAESMassAirList(List<double> CAESMassGasList,
    double GasHeatCapacity, double CombustionChamberTemperature, double
    AirHeatCapacity, double HU, double MAL, double StorageTemperature)
        {
          // CAESMassAir List constructor.
          List<double> CAESMassAirList = new List<double>();

          double CAESMassAir;

          for (int i = 0; i < CAESMassGasList.Count; i++)
          {
            CAESMassAir = (CAESMassGasList[i] * GasHeatCapacity *
    CombustionChamberTemperature) / (AirHeatCapacity * StorageTemperature + (HU
    * 1000 / MAL));
            CAESMassAirList.Add(CAESMassAir);
          }

          return CAESMassAirList;
        }

        #endregion CAES Mass Air List

        #region CAES Mass Fuel List

        /// <summary>
        /// Returns the CAES Mass Fuel List based on the CAES Mass Air List, the CAES
    Mass Gas List, (Kg/sec).
        /// </summary>
        /// <param name="CAESMassAirList">The CAES Mass Air List (Kg/sec).</param>
        /// <param name="CAESMassGasList">The CAES Mass Gas List
    (Kg/sec).</param>
        /// <returns></returns>
        public static List<double> CAESMassFuelList(List<double> CAESMassAirList,
    List<double> CAESMassGasList)
        {
          // CAESMassFuel List constructor.
          List<double> CAESMassFuelList = new List<double>();

          double CAESMassFuel;

          for (int i = 0; i < CAESMassAirList.Count; i++)
          {
            CAESMassFuel = CAESMassGasList[i] - CAESMassAirList[i];
```

```
            CAESMassFuelList.Add(CAESMassFuel);
        }

        return CAESMassFuelList;
    }

    #endregion CAES Mass Fuel List

    #region Air Storage Level List

    /// <summary>
    /// Returns the Air Storage Level List based on the Energy Need List, the Mass
Air Compressor List, the CAES Mass Air List and the Compressor Max Air (Kg).
    /// </summary>
    /// <param name="EnergyNeedList">The Energy Need List (MW).</param>
    /// <param name="MassAirCompressorList">The Mass Air Compressor List
(Kg/Hour).</param>
    /// <param name="CAESMassAirList">The CAES Mass Air List (Kg/sec).</param>
    /// <param name="CompressorMaxAir">The Compressor Max Air (Kg).</param>
    /// <param name="DOD">The DOD.</param>
    /// <returns></returns>
    public static List<double> AirStorageLevelList(List<double> EnergyNeedList,
List<double> MassAirCompressorList, List<double> CAESMassAirList, double
CompressorMaxAir, double DOD)
    {
      // AirStorageLevel List constructor.
      List<double> AirStorageLevelList = new List<double>();

      double AirStorageLevel;
      double DefaultCompressorMaxAirAndDod = CompressorMaxAir * (1 - DOD);

      // For the 1st value of the list.
      AirStorageLevel = DefaultCompressorMaxAirAndDod;
      if (MassAirCompressorList[0] > 0)
      {
        AirStorageLevel += MassAirCompressorList[0];
      }

      AirStorageLevelList.Add(AirStorageLevel);

      // For all the other values of the list (2nd till end).
      for (int i = 1; i < EnergyNeedList.Count; i++)
      {
        if (EnergyNeedList[i] < 0)
        {
          double CAESMassAirList3600Times = CAESMassAirList[i] * 3600;
```

```
            if (AirStorageLevelList[i - 1] - DefaultCompressorMaxAirAndDod >
CAESMassAirList3600Times)
            {
              AirStorageLevel = AirStorageLevelList[i - 1] -
CAESMassAirList3600Times;
            }
            else
            {
              AirStorageLevel = DefaultCompressorMaxAirAndDod;
            }
          }
          else
          {
            double MassAirCompressorPlusPreviousAirStorage =
MassAirCompressorList[i] + AirStorageLevelList[i - 1];
            if (MassAirCompressorPlusPreviousAirStorage > CompressorMaxAir)
            {
              AirStorageLevel = CompressorMaxAir;
            }
            else
            {
              AirStorageLevel = MassAirCompressorPlusPreviousAirStorage;
            }
          }

          AirStorageLevelList.Add(AirStorageLevel);
        }

        return AirStorageLevelList;
      }

      #endregion Air Storage Level List

      #region Air Storage Difference List

      /// <summary>
      /// Returns the Air Storage Difference List based on the Air Storage Level List,
  the Compressor Max Air (Kg) and the DOD.
      /// </summary>
      /// <param name="AirStorageLevelList">The Air Storage Level List
  (Kg).</param>
      /// <param name="CompressorMaxAir">The Compressor Max Air (Kg).</param>
      /// <param name="DOD">The DOD.</param>
      /// <returns></returns>
      public static List<double> AirStorageDifferenceList(List<double>
  AirStorageLevelList, double CompressorMaxAir, double DOD)
```

```
        {
            // AirStorageDifference List constructor.
            List<double> AirStorageDifferenceList = new List<double>();

            double AirStorageDifference;

            // For the 1st value of the list.
            AirStorageDifference = CompressorMaxAir * (1 - DOD) - AirStorageLevelList[0];

            AirStorageDifferenceList.Add(AirStorageDifference);

            // For all the other values of the list (2nd till end).
            for (int i = 1; i < AirStorageLevelList.Count; i++)
            {
                AirStorageDifference = AirStorageLevelList[i - 1] - AirStorageLevelList[i];
                AirStorageDifferenceList.Add(AirStorageDifference);
            }

            return AirStorageDifferenceList;
        }

        #endregion Air Storage Difference List

        #region MA Actual List

        /// <summary>
        /// Returns the MA Actual List based on the Air Storage Difference List (Kg/sec).
        /// </summary>
        /// <param name="AirStorageLevelList">The Air Storage Difference List
(Kg).</param>
        /// <returns></returns>
        public static List<double> MAActualList(List<double> AirStorageDifferenceList)
        {
            // MAActualList List constructor.
            List<double> MAActualList = new List<double>();

            double MAActual;

            for (int i = 0; i < AirStorageDifferenceList.Count; i++)
            {
                if (AirStorageDifferenceList[i] > 0)
                {
                    MAActual = AirStorageDifferenceList[i] / 3600;
                }
                else
                {
```

```
            MAActual = 0;
        }

        MAActualList.Add(MAActual);
    }

    return MAActualList;
}

#endregion MA Actual List

#region Dual Turbine Power List

/// <summary>
/// Returns the Dual Turbine Power List based on the MA Actual List, the MAL,
the Combustion Chamber Temperature, the Gas Heat Capacity, the Expansion Ratio,
the GPower and the CPTurbine (MW).
/// </summary>
/// <param name="MAActualList">The MA Actual List (Kg/sec).</param>
/// <param name="MAL">The MAL.</param>
/// <param name="CombustionChamberTemperature">The Combustion
Chamber Temperature (Kelvin).</param>
/// <param name="GasHeatCapacity">The Gas Heat Capacity
(KJ/Kg*Kelvin).</param>
/// <param name="ExpansionRatio">The Expansion Ratio.</param>
/// <param name="GPower">The GPower.</param>
/// <param name="CPTurbine">The CPTurbine.</param>
/// <returns></returns>
public static List<double> DualTurbinePowerList(List<double> MAActualList,
double MAL, double CombustionChamberTemperature, double GasHeatCapacity,
double ExpansionRatio, double GPower, double CPTurbine)
    {
        // DualTurbinePower List constructor.
        List<double> DualTurbinePowerList = new List<double>();

        double DualTurbinePower;

        for (int i = 0; i < MAActualList.Count; i++)
        {
            DualTurbinePower = MAActualList[i] * (1+1/MAL) *
CombustionChamberTemperature * GasHeatCapacity * (1-
1/(Math.Pow(ExpansionRatio, GPower))) / 1000 * CPTurbine;
            DualTurbinePowerList.Add(DualTurbinePower);
        }

        return DualTurbinePowerList;
```

```
    }

    #endregion Dual Turbine Power List

    #region Final Energy Need List

    /// <summary>
    /// Returns the Final Energy List based on the Power Production List, the Energy
Consumption List and the Dual Turbine Power List (MW).
    /// </summary>
    /// <param name="PowerProductionList"> The Power Production List
(MW).</param>
    /// <param name="EnergyConsumptionList">The Energy Consumption List
(MW).</param>
    /// <param name="DualTurbinePowerList">The Dual Turbine Power List
(MW).</param>
    /// <returns></returns>
    public static List<double> FinalEnergyNeedList(List<double>
PowerProductionList, List<double> EnergyConsumptionList, List<double>
DualTurbinePowerList)
    {
      // FinalEnergyNeed List constructor.
      List<double> FinalEnergyNeedList = new List<double>();

      double FinalEnergyNeed;

      for (int i = 0; i < PowerProductionList.Count; i++)
      {
        double DualTurbinePowerPlusPowerProductionMinusEnergyConsumption =
DualTurbinePowerList[i] + PowerProductionList[i] - EnergyConsumptionList[i];
        if (DualTurbinePowerPlusPowerProductionMinusEnergyConsumption < 0)
        {
          FinalEnergyNeed = -
DualTurbinePowerPlusPowerProductionMinusEnergyConsumption;
        }
        else
        {
          FinalEnergyNeed = 0;
        }

        FinalEnergyNeed = Math.Round(FinalEnergyNeed, 3);
        FinalEnergyNeedList.Add(FinalEnergyNeed);
      }


      return FinalEnergyNeedList;
```

```
        }

        #endregion Final Energy Need List

        #endregion List Values

        #region Sum Values

        #region Final Energy Need Sum

        /// <summary>
        /// Returns the Final Energy Need Sum (MW).
        /// </summary>
        /// <param name="FinalEnergyNeedList">The Final Energy Need List
(MW).</param>
        /// <returns></returns>
        public static double FinalEnergyNeedSum(List<double> FinalEnergyNeedList)
        {
            double FinalEnergyNeedSum = 0;

            foreach (double FinalEnergyNeed in FinalEnergyNeedList)
            {
                FinalEnergyNeedSum += FinalEnergyNeed;
            }

            return FinalEnergyNeedSum;
        }

        #endregion Final Energy Need Sum

        #region CAES Non Coverage Hours

        /// <summary>
        /// Returns the CAES Non Coverage Hours (Hours).
        /// </summary>
        /// <param name="FinalEnergyNeedList">The Final Energy Need List
(MW).</param>
        /// <returns></returns>
        public static double CAESNonCoverageHours(List<double> FinalEnergyNeedList)
        {
            double CAESNonCoverageHours = 0;

            foreach (double FinalEnergyNeed in FinalEnergyNeedList)
            {
                if (FinalEnergyNeed > 0.001)
                {
```

```
            CAESNonCoverageHours++;
          }
        }

        return CAESNonCoverageHours;
      }

      #endregion CAES Non Coverage Hours

      #region Annual CAES Mass Fuel

      /// <summary>
      /// Returns the Annual CAES Mass Fuel (tons).
      /// </summary>
      /// <param name="MAActualList">The MA Actual List (Kg/sec).</param>
      /// <param name="MAL">The MAL.</param>
      /// <returns></returns>
      public static double AnnualCAESMassFuel(List<double> MAActualList, double
MAL)
      {
        double AnnualCAESMassFuel;
        double MAActualSum = 0;

        foreach (double MAActual in MAActualList)
        {
          MAActualSum += MAActual;
        }

        //AnnualCAESMassFuel = MAActualSum / (MAL * 3.6);
        AnnualCAESMassFuel = (MAActualSum * 3.6) / MAL;

        return AnnualCAESMassFuel;
      }

      #endregion Annual CAES Mass Fuel

      #region Annual Dual Mass Fuel

      /// <summary>
      /// Returns the Annual Dual Mass Fuel (tons).
      /// </summary>
      /// <param name="FinalEnergyNeedSum">The Final Energy Need Sum
(MW).</param>
      /// <param name="HU">The HU.</param>
      /// <param name="CPBrayton">The CPBrayton.</param>
      /// <returns></returns>
```

```
    public static double AnnualDualMassFuel(double FinalEnergyNeedSum, double
HU, double CPBrayton)
    {
      double AnnualDualMassFuel;

      AnnualDualMassFuel = (FinalEnergyNeedSum * 3.6) / (HU * CPBrayton); //
(3.6*1000)/1000

      return AnnualDualMassFuel;
    }

    #endregion Annual Dual Mass Fuel

    #region Annual Total Mass Fuel

    /// <summary>
    /// Returns the Annual Total Mass Fuel (tons).
    /// </summary>
    /// <param name="AnnualCAESMassFuel">The Annual CAES Mass Fuel
(tons).</param>
    /// <param name="AnnualDualMassFuel">The Annual Dual Mass Fuel
(tons).</param>
    /// <returns></returns>
    public static double AnnualTotalMassFuel(double AnnualCAESMassFuel, double
AnnualDualMassFuel)
    {
      double AnnualTotalMassFuel;

      AnnualTotalMassFuel = AnnualCAESMassFuel + AnnualDualMassFuel;

      return AnnualTotalMassFuel;
    }

    #endregion Annual Total Mass Fuel

    #region Annual Brayton Mass Fuel

    /// <summary>
    /// Returns the Annual Brayton Mass Fuel (tons).
    /// </summary>
    /// <param name="EnergyNeedList">The Energy Need List (MW).</param>
    /// <param name="HU">The HU.</param>
    /// <param name="CPBrayton">The CPBrayton.</param>
    /// <returns></returns>
    public static double AnnualBraytonMassFuel(List<double> EnergyNeedList,
double HU, double CPBrayton)
```

```
    {
        double AnnualBraytonMassFuel;
        double EnergyNeedSum = 0;

        foreach (double EnergyNeed in EnergyNeedList)
        {
            if (EnergyNeed < 0)
            {
                EnergyNeedSum += EnergyNeed;
            }
        }

        AnnualBraytonMassFuel = Math.Abs(EnergyNeedSum) * 3.6 / (HU *
CPBrayton); // (3.6*1000)/1000

        return AnnualBraytonMassFuel;
    }

    #endregion Annual Brayton Mass Fuel

    #region Wind Power Excess

    /// <summary>
    /// Returns the Wind Power Excess (MW).
    /// </summary>
    /// <param name="EnergyRejectionList">The Energy Rejection List
(MW).</param>
    /// <returns></returns>
    public static double WindPowerExcess(List<double> EnergyRejectionList)
    {
        double WindPowerExcess = 0;

        foreach (double EnergyRejection in EnergyRejectionList)
        {
            WindPowerExcess += EnergyRejection;
        }

        return WindPowerExcess;
    }

    #endregion Wind Power Excess

    #region Heat Rate

    /// <summary>
    /// Returns the Heat Rate (KWhng/KWhe).
```

```
    /// </summary>
    /// <param name="DualTurbinePowerList">The Dual Turbine Power List
(MW).</param>
    /// <param name="AnnualCAESMassFuel">The Annual CAES Mass Fuel
(tons)</param>
    /// <param name="HU">The HU.</param>
    /// <returns></returns>
    public static double HeatRate(List<double> DualTurbinePowerList, double
AnnualCAESMassFuel, double HU)
    {
      double HeatRate;
      double DualTurbinePowerSum = 0;

      foreach (double DualTurbinePower in DualTurbinePowerList)
      {
        DualTurbinePowerSum += DualTurbinePower;
      }

      HeatRate = AnnualCAESMassFuel * HU / (3.6 * DualTurbinePowerSum);

      return HeatRate;
    }

    #endregion Heat Rate

    #region DMF

    /// <summary>
    /// Returns the DMF (tons).
    /// </summary>
    /// <param name="AnnualBraytonMassFuel"> The Annual Brayton Mass Fuel
(tons).</param>
    /// <param name="AnnualTotalMassFuel">The Annual Total Mass Fuel
(tons).</param>
    /// <returns></returns>
    public static double DMF(double AnnualBraytonMassFuel, double
AnnualTotalMassFuel)
    {
      double DMF;

      DMF = AnnualBraytonMassFuel - AnnualTotalMassFuel;

      return DMF;
    }

    #endregion DMF
```

```
        #region Wind Calms

        /// <summary>
        /// Returns a Wind Calms Object based on the Wind Speed List Containing two
    variables. The Wind Calms Total Counter and the Max Concecutive Wind Calms.
        /// </summary>
        /// <param name="WindSpeedList">The Wind Speed List (m/s).</param>
        /// <returns></returns>
        public static WindCalmsObject WindCalms(List<double> WindSpeedList)
        {
          double WindCalmsTotalCounter = 0;
          double MaxConcecutiveWindCalms = 0;
          double ConcecutiveWindCalms = 0;

          foreach (double windspeed in WindSpeedList)
          {
            if (windspeed < 4)
            {
              WindCalmsTotalCounter += 1;
              ConcecutiveWindCalms += 1;
            }
            else
            {
              if (ConcecutiveWindCalms > MaxConcecutiveWindCalms)
              {
                MaxConcecutiveWindCalms = ConcecutiveWindCalms;
              }
              ConcecutiveWindCalms = 0;
            }
          }

          WindCalmsObject WindCalms = new
    WindCalmsObject(WindCalmsTotalCounter, MaxConcecutiveWindCalms);

          return WindCalms;
        }

        #endregion Wind Calms

        #endregion Sum Values

        #region Wind Calms Class

        /// <summary>
        /// The Wind Calms Object.
```

```
    /// </summary>
    public class WindCalmsObject
    {
      public double WindCalmsTotalCounter { get; set; }
      public double MaxConcecutiveWindCalms { get; set; }
      public WindCalmsObject(double windCalmsTotalCounter, double
maxConcecutiveWindCalms)
      {
        WindCalmsTotalCounter = windCalmsTotalCounter;
        MaxConcecutiveWindCalms = maxConcecutiveWindCalms;
      }
    }

    #endregion Wind Calms Class
  }
}
```