ΕΘΝΙΚΟ ΜΕΤΣΟΒΙΟ ΠΟΛΥΤΕΧΝΕΙΟ
ΣΧΟΛΗ ΜΗΧΑΝΟΛΟΓΩΝ ΜΗΧΑΝΙΚΩΝ
ΤΟΜΕΑΣ Μ.Κ. & Α.Ε.
Εργαστήριο Αυτομάτου Ελέγχου

Εργασία για το μάθημα: Ηλεκτρομηχανικά Συστήματα Μετατροπής Ενέργειας

Πειραματική εκτίμηση τεχνικών χαρακτηριστικών κινητήρα Σ.Ρ.

Ημερομηνία Ανάθεσης:	15 Δεκεμβρίου 2020
Διδάσκοντες:	Ε.Γ. Παπαδόπουλος, Καθηγητής
	Ι.Ν. Νταβλιάκος, ΕΔΙΠ

Κωστόπουλος Γεώργιος του Αργυρίου
Αριθμός Μητρώου: mc19116

Αθήνα, 22 Ιανουαρίου 2021

Περίληψη

Στη παρούσα εργασία γίνεται μία πειραματική εκτίμηση των παραμέτρων ενός κινητήρα Σ.Ρ. Μ.Μ με χρήση μόνο ενός μικροελεκτή, ενός ρελέ και μίας αντίστασης. Αρχικά, εφαρμόζεται στον κινητήρα μία βηματική διέγερση τάσης και γίνονται μετρήσεις του ρεύματος του i_a. Στην συνέχεια γίνεται παλινδρόμηση των μετρήσεων στη θεωρητική καταστατική εξίσωση του κινητήρα και προσδιορίζονται οι παράμετροι του.

Περιεχόμενα

Κατάλογος Σχημάτων

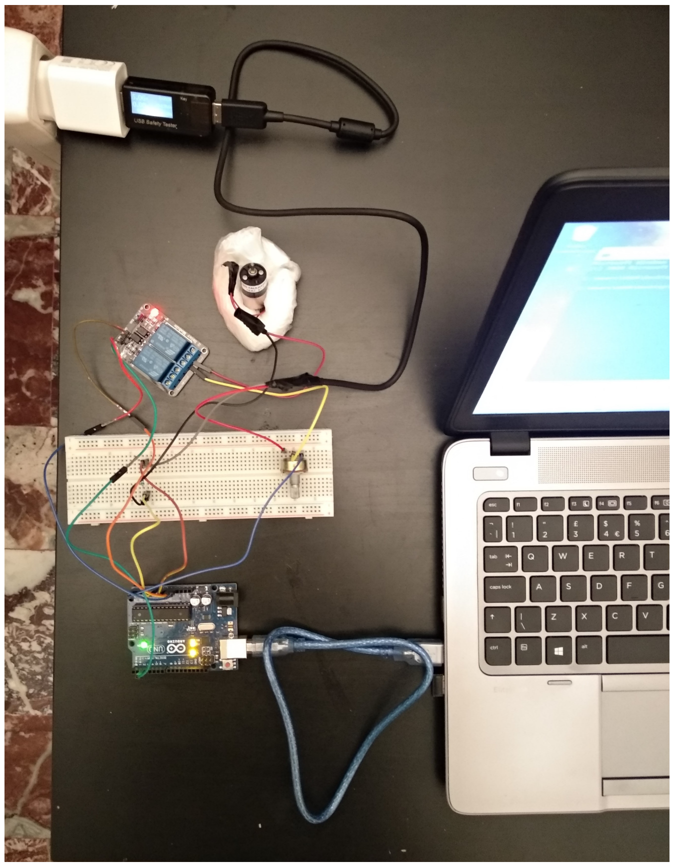
Κατάλογος Πινάκων

1 Εισαγωγή

Για την εύρεση των χαρακτηριστικών του κινητήρα στη παρούσα εργασία γίνονται πειράματα προσδιορισμού της απόκρισης του ρεύματος σε βηματική διέγερσης τάσης και πραγματοποιείται παλινδρόμηση των μετρούμενων δεδομένων στην καταστατική εξίσωση του κινητήρα. Για μεγαλύτερη αξιοπιστία των μετρήσεων το πείραμα έλαβε χώρα 5 φορές συνολικά.

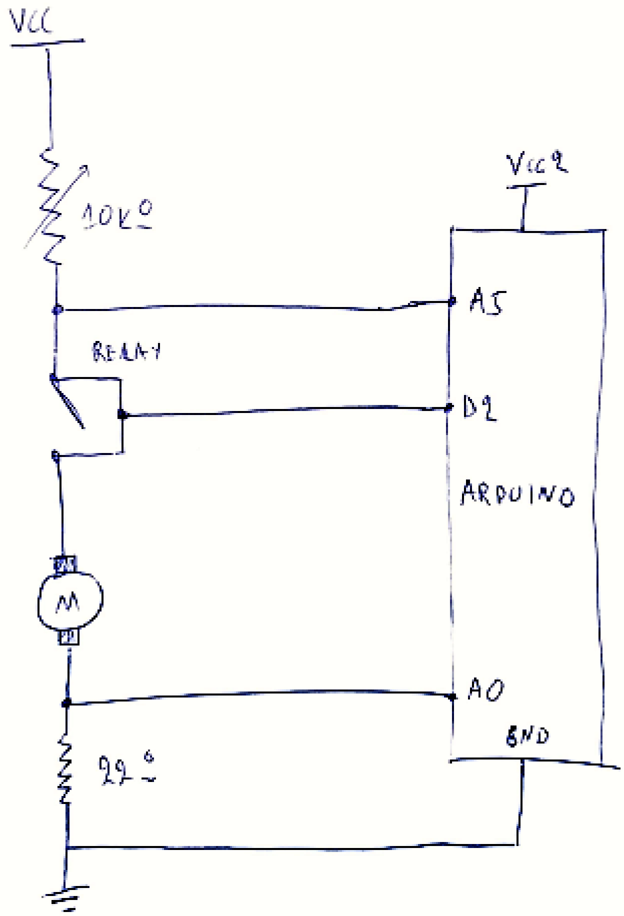
2 Περιγραφή Πειραματικής Διάταξης

Για την μέτρηση του ρεύματος του κινητήρα συνάρτηση του χρόνου (απόκριση στο πεδίο του χρόνου) και της τάσης του (είσοδος) στήθηκε η πειραματική διάταξη που απεικονίζεται στην εικόνα 1 και διαγραμματικά στο σχήμα 2.



Σχήμα 1: Φωτογραφία πειραματικής διάταξης

Η πηγή τάσης DC των 5 V είναι συνδεδεμένη σε σειρά με ένα ποτενσιόμετρο, ένα ρελέ, τον κινητήρα και μία αντίσταση των 22 Ω η οποία χρησιμοποιείται για λόγους προστασίας του μικροελεκτή. Όταν το ρελέ κλείσει, ο κινητήρας τίθεται σε λειτουργία. Για την μέτρηση του ρεύματος του κινητήρα, μετριέται η τάσης στα άκρα της αντίστασης των 22 Ω με τη χρήση ενός μικροελεγκτή Arduino. Ακόμα, ο μικροελεκτής κλείνει το ρελέ και μετράει την τάση εισόδου η οποία μεταβάλλεται απο το ποτενσιόμετρο.
Στην εκκίνηση του πειράματος, μετά απο 2 δευτερόλεπτα περίπου κλείνει το ρελέ και ο κινητήρας μπαίνει σε λειτουργία. Αφήνεται για περίπου 5 δευτερόλεπτα ώστε να επέλθει στη μόνιμη κατάσταση και έπειτα αρχίζει μία τυχαία μεταβολή της τάσης εισόδου η οποία λαμβάνει χώρα μέχρι το πέρας του πειράματος.

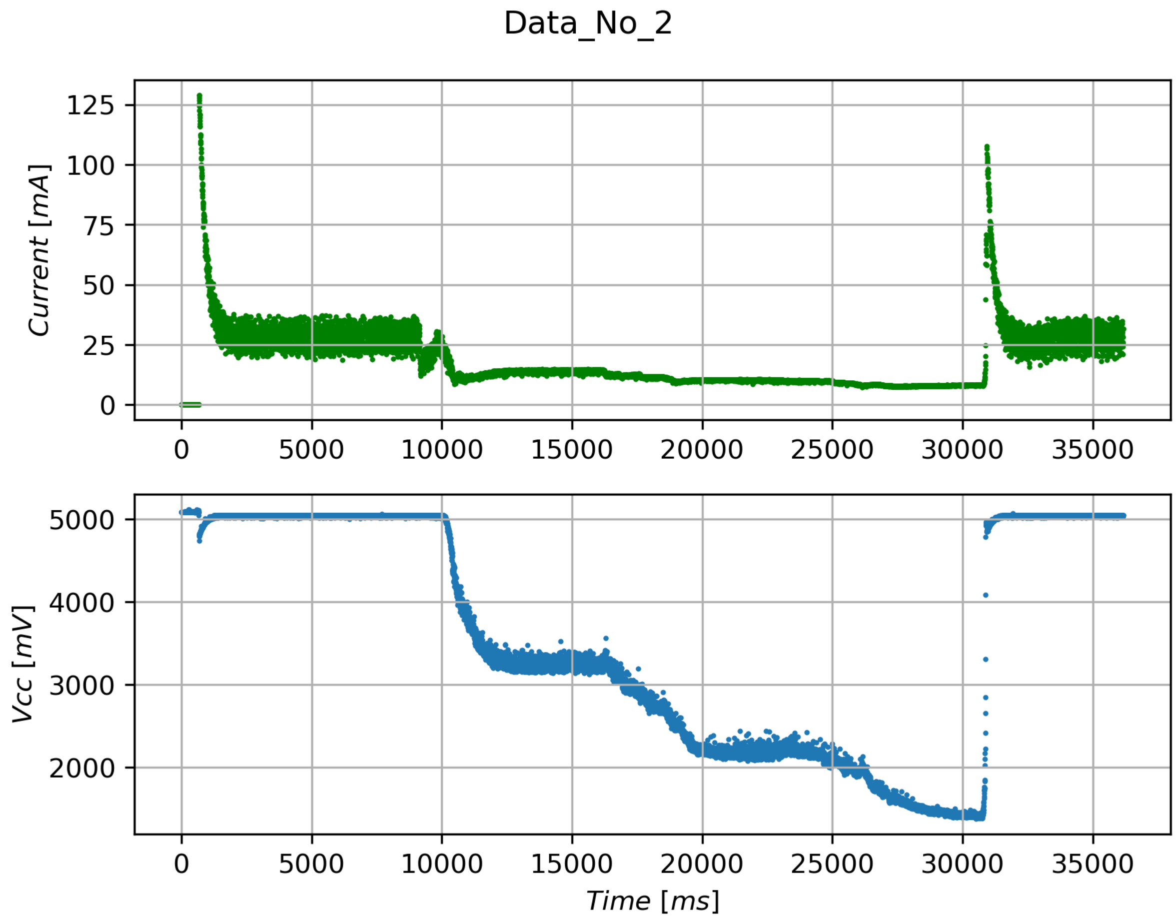


Σχήμα 2: Διάγραμμα πειραματικής διάταξης

Για τη μέτρηση της τάσης με σχετικά καλύτερη ακρίβεια γίνεται,μέσω κατάλληλων εντολών, χρήση της εσωτερικής τάσης αναφοράς 1.1 V του μικροελεκτή.
Ο κινητήρας βρίσκεται πάνω σε μία πλαστική σακούλα, με σκοπό των περιορισμό των πολλών δονήσεων που παράγει οι οποίες εισάγουν θόρυβο στις μετρήσεις.

3 Επεξεργασία Μετρήσεων

Οι μετρήσεις που τελικά λαμβάνονται απο τον μικροελεκτή μεταφέρονται στον υπολογιστή απο τη σειριακή θύρα επικοινωνίας, μέσω ενός script σε γλώσσα python , και αποθηκεύονται σε αρχείο τύπο .csv. Αυτές αφορούν τον χρόνο, την τάση εισόδου του κινητήρα και τη τάση στα άκρα τη αντίστασης των 22 Ω, όπως φαίνονται στο σχήμα 3.



Σχήμα 3: Μετρούμενο ρεύμα και τάση εισόδου του κινητήρα

4 Προσδιορισμός Τεχνικών Χαρακτηριστικών

Απο ΝΤΚ στο ισοδύναμο ηλεκτρικό κύκλωμα του κινητήρα προκύπτει η σχέση

$$v_s = L_a \frac{di_a}{dt} + R_a i_a + K_t \omega \tag{1}$$

ενώ απο τον νόμο του Νεύτωνα για περιστροφική κίνηση

$$T = K_t i_a = J\frac{d\omega}{dt} + B\omega + T_d \tag{2}$$

όπου v_s η τάση εισόδου του κινητήρα, i_a το ρεύμα του δρομέα, R_a η εσωτερική του αντίσταση, K_t η σταθερά ροπής του, ω οι στροφές του άξονα, J η ροπή αδράνειας και B συντελεστής δυναμικής τριβής. Η σταθερά T_d εκφράζει τη ροπή διαταραχής και είναι ο όρος που περιλαμβάνει την στατική τριβή. Επειδή εισάγει στο σύστημα μη γραμμικότητα [2], θα θεωρηθεί $T_d = 0$.
Οι διαφορικές εξισώσεις 1 και 2 γράφονται σε μορφή εξισώσεων κατάστασης [1] ως

$$\begin{bmatrix} \dot{i}_a \\ \dot{\omega} \end{bmatrix} = \begin{bmatrix} \frac{-R_a}{L_a} & \frac{-K_t}{L_a} \\ \frac{K_t}{J} & \frac{-B}{J} \end{bmatrix} \begin{bmatrix} i_a \\ \omega \end{bmatrix} + \begin{bmatrix} \frac{1}{L_a} \\ 0 \end{bmatrix} v_s \tag{3}$$

$$y = \begin{bmatrix} 1 & 0 \end{bmatrix} \begin{bmatrix} i_a \\ \omega \end{bmatrix} + 0 \tag{4}$$

Στη μόνιμη κατάσταση, απο τη σχέση 1 προκύπτει

$$K_t = \frac{v_{s,ss} - R_a i_{a,ss}}{\omega_{ss}} \tag{5}$$

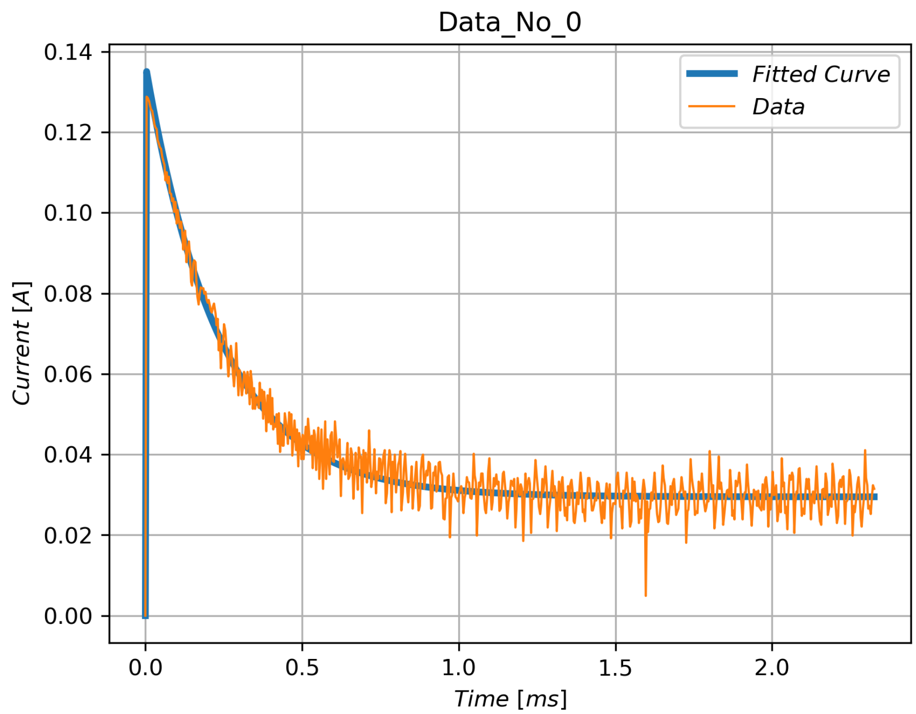
και απο τη σχέση 2

$$B = \frac{K_t i_{a,ss}}{\omega_{ss}} \tag{6}$$

Οι τιμές των ω_{ss} και $i_{a,ss}$ είναι εύκολα μετρήσιμες. Στη παρούσα διάταξη το ρεύμα κενού μετριέται, αλλά οι στροφές απαιτούν για την μέτρηση τους ένα ταχύμετρο ή έναν αισθητήρα ταχύτητας περιστροφής. Τελικά οι παράμετροι R_a, ω_{ss}, J, L υπολογίζονται απο παλινδρόμηση στα δεδομένα, όπως φαίνεται στο σχήμα 4.
Για την παλινδρόμηση με τη μέθοδο των ελαχίστων τετραγώνων, δημιουργείται μία συνάρτηση σφάλματος η οποία περιέχει τη διαφορά των μετρούμενων τιμών και αυτών που προέρχονται απο την επίλυση του συστήματος. Η αριθμητική επίλυση των σχέσεων 3 και 4 γίνεται μέσω της βιβλιοθήκης scipy [3] και σαν είσοδος θεωρείται ο μέσος όρος της V_{cc} για χρόνο απο 3 εως 8 sec.
Η παραπάνω διαδικασία έγινε και για τα 5 σετ μετρήσεων και στον πίνακα 1 παρουσιάζονται οι παράμετροι που προέκυψαν
Παρατηρούμε οτι υπάρχει μικρή διακύμανση μεταξύ των εκτιμώμενων παραμέτρων του κάθε πειράματος. Με γνωστές τις παραμέτρους πλεόν, γίνεται επίλυση των σχέσεων 3 και 4 σε όλο το χρονικό διάστημα των μετρήσεων, με είσοδο τις μετρούμενες τιμές της τάσης εισόδου και στο διάγραμμα 5 συγκρίνονται τα αποτελέσματα με τις μετρούμενες τιμές.
Τέλος, επιλέγοντας για έξοδο του συστήματος την γωνιακή ταχύτητα, η σχέση 4 γίνεται

$$y = \begin{bmatrix} 0 & 1 \end{bmatrix} \begin{bmatrix} i_a \\ \omega \end{bmatrix} + 0 \tag{7}$$

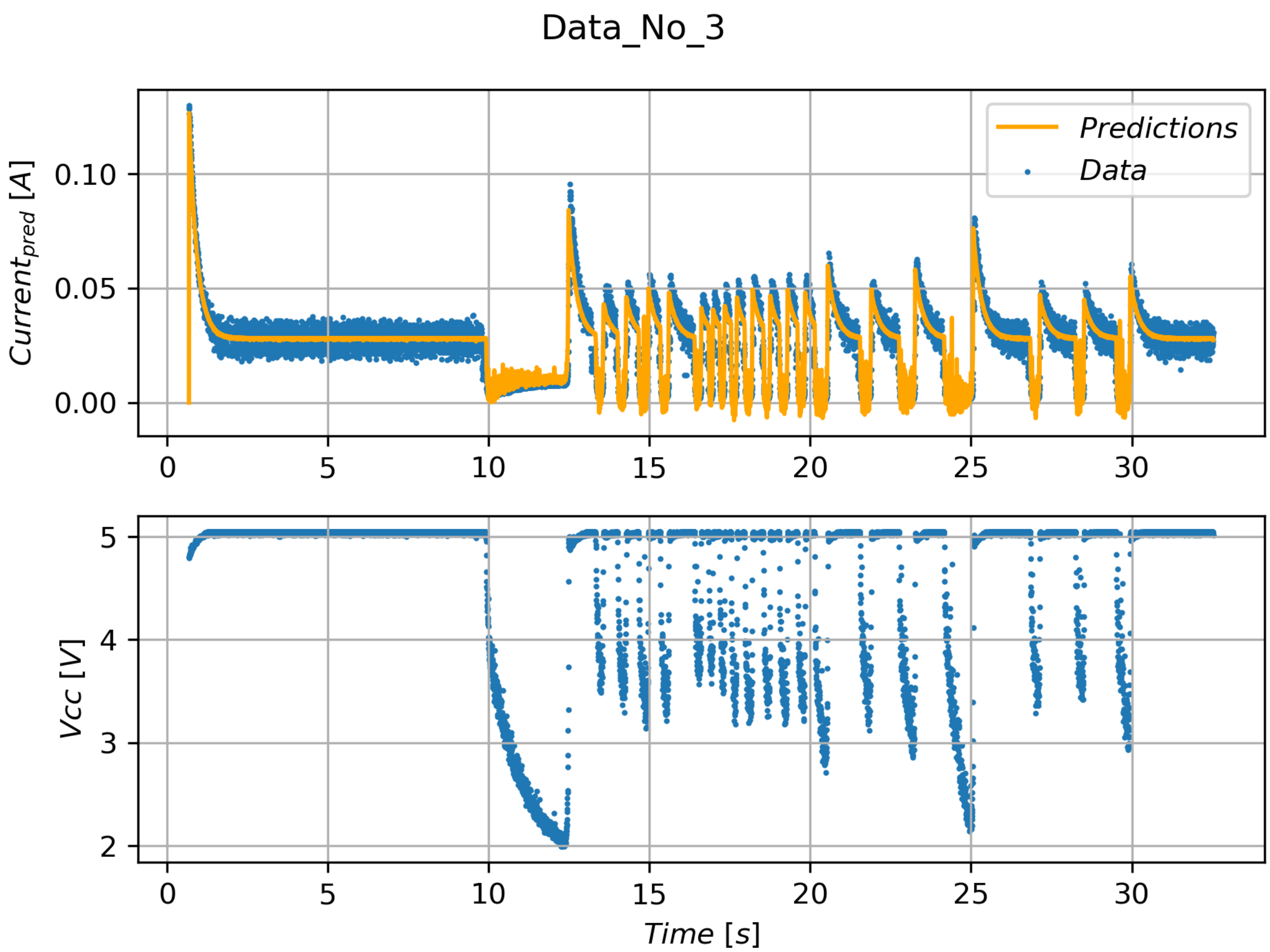
και απο την επίλυση του συστήματος προκύπτει το διάγραμμα 6



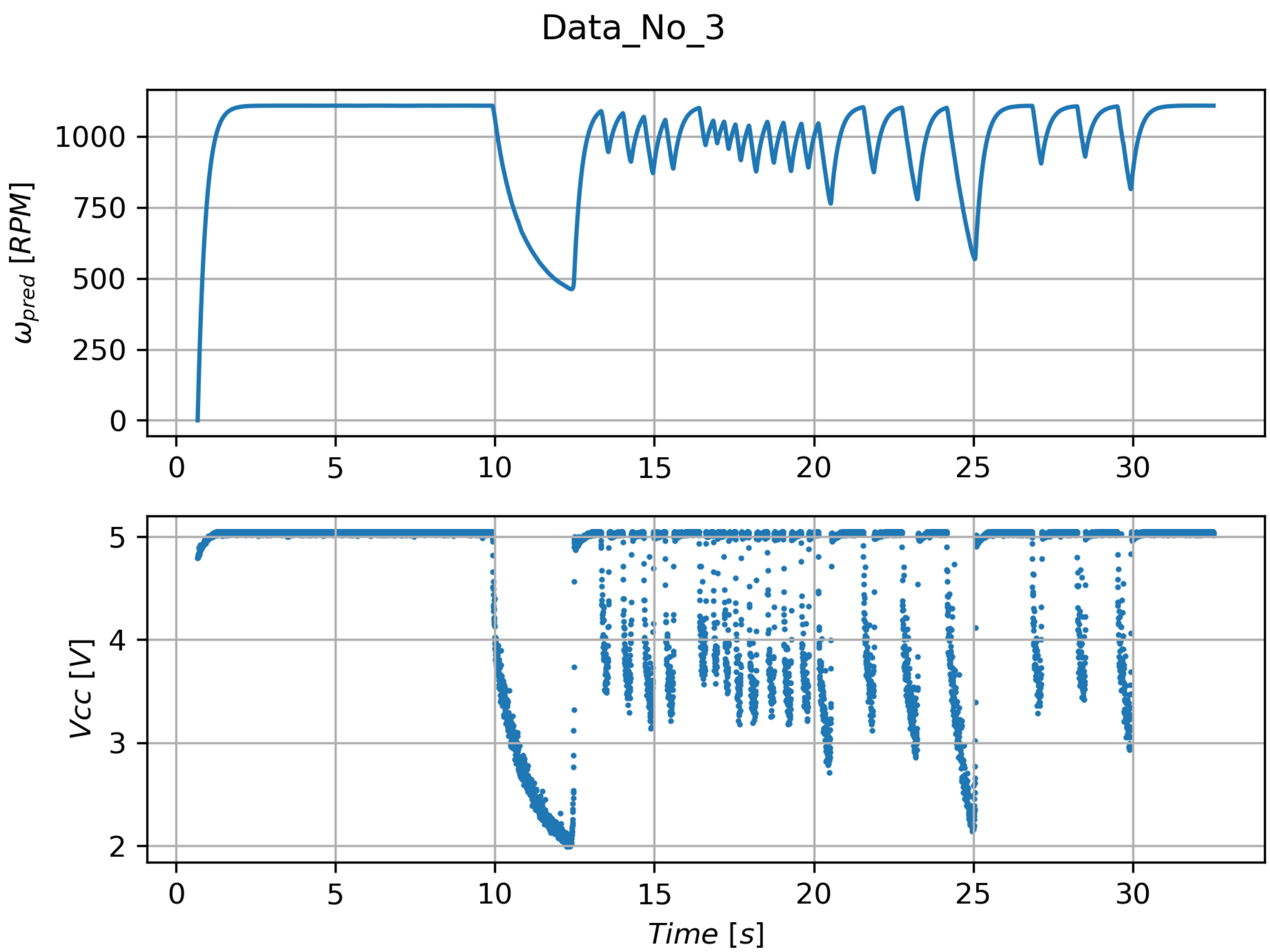
Σχήμα 4: Μη γραμμική παλινδρόμηση των μετρήσεων

Πίνακας 1: Παράμετροι για κάθε ένα απο τα 5 σετ μετρήσεων

A/A	R_a $[\Omega]$	ω_{ss} $[RPM]$	J $[kgm^2]$	L $[H]$
1	3.715×10^1	1.125×10^3	1.031×10^{-7}	9.863×10^{-11}
2	3.684×10^1	1.069×10^3	1.076×10^{-7}	9.400×10^{-11}
3	3.732×10^1	1.141×10^3	1.002×10^{-7}	9.629×10^{-11}
4	3.731×10^1	1.108×10^3	1.030×10^{-7}	9.490×10^{-11}
5	3.743×10^1	1.153×10^3	9.740×10^{-8}	9.888×10^{-11}



Σχήμα 5: Θεωρητική και πειραματική απόκριση ρέυματος για διάφορες τάσεις εισόδου



Σχήμα 6: Θεωρητική απόκριση γωνιακής ταχύτητας για διάφορες τάσεις εισόδου

5 Συμπεράσματα

5.1 Συμπεράσματα

Η μέθοδος των ελαχίστων τετραγώνων δίνει συνεπή και λογικά αποτελέσματα για τις εκτιμώμενες παραμέτρους του κινητήρα. Ένα απο τα μειονεκτήματα της είναι πως έχει μεγάλη ευαισθησία στις αρχικές τιμές των παραμέτρων που πρέπει να εισαχθούν. Ακόμα, ένα μάλλον σοβαρό μειονέκτημα της παρούσας μεθόδου είναι πως θεωρήθηκε οτι η τριβή Coulomb είναι μηδενική, Αυτό δεν ισχύει στην πραγματικότητα, ειδικά στον συγκεκριμένο κινητήρα όπου υπάρχει ενσωματωμένος στον άξονα του μειωτήρας στροφών. Επίσης, παρατηρούμε οτι η αυτεπαγωγή του κινητήρα είναι πολύ μικρή και μπορεί να θεωρηθεί αμελητέα, απλοποιώντας το σύστημα χωρίς όμως να βοηθάει κάπου ουσιαστικά τη μέθοδο.

5.2 Μελλοντική Εργασία

Ένας τρόπος υπέρβασης του πρώτου μειονεκτήματος, είναι η ανάπτυξη ενός π.χ. γενετικού αλγορίθμου που με στοχαστικές διαδικασίες σε σχετικά λίγες επαναλήψεις θα μπορεί να προσδιορίσει τις αρχικές προβλέψεις που κάνουν τον αλγόριθμο των ελαχίστων τετραγώνων να συγκλίνει. Έτσι επιτυγχάνεται ένας συστηματικός τρόπος εκτίμησης των παραμέτρων για διάφορες κατηγορίες κινητήρων χωρίς να χρειάζεται η επέμβαση του χρήστη. Για τη στατική τριβή, πρέπει είτε να επιλυθεί το μη γραμμικό μοντέλο είτε να αναπτυχθεί ένα γραμμικό που να την εμπεριέχει.

Με ένα πολύμετρο μετριέται εύκολα η αντίσταση του κινητήρα και με ένα ταχύμετρο η γωνιακή ταχύτητα του στη μόνιμη κατάσταση. Έτσι απομένουν 2 παράμετροι που πρέπει να προσδιοριστούν κάνοντας πιο εύκολη την ζωή του αλγορίθμου των ελαχίστων τετραγώνων αλλά κυρίως μειώνοντας την αβεβαιότητα που εισέρχεται στο σύστημα.

Για να θεωρηθεί πιο ολοκληρωμένη η μελέτη αυτή , πρέπει να εξεταστεί και η μετάδοση των σφαλμάτων μέσα στο σύστημα. Ο μικροελεκτής έχει κάποιο σφάλμα στις μετρήσεις του και απο τη παλινδρόμηση προκύπτει ένας πίνακας συνδιακύμανσης των παραμέτρων. Αυτά θα πρέπει να συνυπολογιστούν ώστε να προκύψουν κάποια διαστήματα εμπιστοσύνης για τις τιμές των εξόδων του συστήματος.

Βιβλιογραφία

[1] Ε. Παπαδόπουλος. *Ηλεκτρομηχανικά Συστήματα Μετατροπής Ενέργειας*. Αθήνα, 2010.

[2] M. Ruderman, J. Krettek, F. Hoffmann, and T. Bertram. Optimal state space control of dc motor. *IFAC Proceedings Volumes*, 41(2):5796 – 5801, 2008. 17th IFAC World Congress.

[3] P. Virtanen, R. Gommers, T. E. Oliphant, M. Haberland, T. Reddy, D. Cournapeau, E. Burovski, P. Peterson, W. Weckesser, J. Bright, S. J. van der Walt, M. Brett, J. Wilson, K. Jarrod Millman, N. Mayorov, A. R. J. Nelson, E. Jones, R. Kern, E. Larson, C. Carey, İ. Polat, Y. Feng, E. W. Moore, J. Vand erPlas, D. Laxalde, J. Perktold, R. Cimrman, I. Henriksen, E. A. Quintero, C. R. Harris, A. M. Archibald, A. H. Ribeiro, F. Pedregosa, P. van Mulbregt, and S. . . Contributors. SciPy 1.0: Fundamental Algorithms for Scientific Computing in Python. *Nature Methods*, 17:261–272, 2020.

ΠΑΡΑΡΤΗΜΑ

Α′ Κώδικας

Κώδικας 1: Κύριο πρόγραμμα

```
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
from scipy.signal import savgol_filter
from scipy.optimize import curve_fit
from scipy import signal
from scipy import optimize
from proccess_data_functions import *

Rcontrol = 22 # Ohms

Ra_ls,omega_ss_ls,J_ls,L_ls = (np.empty(5) for i in range(4))
for i in range(5):
    name = 'Datasets\\measurements'+str(i)+'.csv'
    df = pd.read_csv(name,error_bad_lines=False)
    df['Current'] = df.Voltage/Rcontrol
    Vmax = df.Voltage.max()

    title = 'Data_No_{}'.format(i)

    plot_Vcc(df,title)

    plot_Voltage(df,title)

    plot_Current_and_Vcc(df,title)

    tmin = 3e3
    tmax = 8e3
    Vcc_SS = Vcc_ss(df,tmin,tmax)
    i_ss = Voltage_ss(df,tmin,tmax)/Rcontrol

    y_data = df.Current[(df.Time<=3e3)].iloc[df.Current.idxmax()-1:].to_numpy()
    x_data = df.Time[(df.Time<=3e3)].iloc[df.Current.idxmax()-1:].to_numpy(dtype='float64')
    x_data = [(i - x_data.min())*1e-3 for i in x_data]

    ### Curve fit
    def f_resid(p):
        """
        function to pass to optimize.leastsq
        The routine will square and sum the values returned by
        this function
        """
        Kt = (Vcc_SS - p[0]*i_ss)/p[1]
        Beq = (Kt*i_ss)/p[1]
        A = np.array([[-p[0]/p[-1],-Kt/p[-1]],
                      [Kt/p[2],-Beq/p[2]]])
        B = np.array([[1/p[-1]],
                      [0]])
        C = np.array([1,0])
        D = 0.0
        syst = signal.StateSpace(A,B,C,D)
        tout, yout, xout = signal.lsim2(syst, U=np.ones(len(x_data))*Vcc_SS, T=x_data)
        return y_data-yout

    initial_guess =(38,1e3,1e-7,1e-10) #initial guess for params
    sol = optimize.leastsq(f_resid, initial_guess)


    Ra, omega_ss, J, L = sol[0]
    Kt = (Vcc_SS - Ra*i_ss)/omega_ss
    Beq = (Kt*i_ss)/omega_ss
    A = np.array([[-Ra/L,-Kt/L],
```

```
                  [Kt/J,-Beq/J]])
    B = np.array([[1/L],
                  [0]])
    C = np.array([1,0])
    D = 0.0
    syst = signal.StateSpace(A,B,C,D)
    tout, yout, xout = signal.lsim2(syst, U=np.ones(len(x_data))*Vcc_SS, T=x_data)

    plot_fitted_curve(tout,yout,x_data,y_data,title)

    filename = 'results_data'+str(i)+'.txt'
    Results(Ra,omega_ss,J,Kt,Beq,L,title).print_
    Results(Ra,omega_ss,J,Kt,Beq,L,title).save_(filename)

    Ra_ls[i] = Ra
    omega_ss_ls[i] = omega_ss
    J_ls[i] = J
    L_ls[i] = L


    ### ia response to all data
    #df = df[(df.Time<=1e3)]
    t = 1e-3*df.Time.iloc[df.Current.idxmax():]
    u = df.VoltageVcc.iloc[df.Current.idxmax():]
    y = df.Current.iloc[df.Current.idxmax():]
    tout, yout, xout = signal.lsim2(syst, U=u.to_numpy(),
                                    T=t.to_numpy())

    plot_Current_predicted(tout,yout,t,y,title)

    plot_Current_predicted_and_Vcc(tout,yout,t,y,u,title)

    ### omega response to all data
    C = np.array([0,1])
    syst = signal.StateSpace(A,B,C,D)
    tout, yout, xout = signal.lsim2(syst,U=u.to_numpy(),
                                    T=t.to_numpy())

    plot_omega_predicted(tout,yout,title)

    plot_omega_predicted_and_Vcc(tout,yout,t,u,title)

    #break
```

Κώδικας 2: Βοηθητικές κλάσεις και συναρτήσεις

```
import matplotlib.pyplot as plt
import sys

class Results():
    def __init__(self,Ra,omega_ss,J,Kt,Beq,L=None,title=None):
        self.Ra = Ra
        self.omega_ss = omega_ss
        self.J = J
        self.Kt = Kt
        self.Beq = Beq
        self.L = L
        self.title = title

    @property
    def print_(self):
        if not self.title==None:
            print('### {} ###'.format(self.title))
        print('Ra = {:.3E} Ohms'.format(self.Ra))
        print('omega ss = {:.3E} RPM'.format(self.omega_ss))
        print('J = {:.3E} kg m2'.format(self.J))
        if not self.L==None:
            print('L = {:.3E} H'.format(self.L))
        print('Kt = {:.3E} N m / A'.format(self.Kt))
        print('Beq = {:.3E} N m s'.format(self.Beq))
        print('### ---- ###')
```

```
    def save_(self,filename):
        original_stdout = sys.stdout
        with open(filename, 'w') as f:
            sys.stdout = f # Change the standard output to the file created.
            self.print_
            sys.stdout = original_stdout # Reset the standard output to its original value
        print('\nfile '+filename+' successfully written\n')

def Vcc_ss(df,tmin,tmax):
    return df.VoltageVcc[(df.Time>=tmin) & (df.Time<=tmax)].mean()

def Voltage_ss(df,tmin,tmax):
    return df.Voltage[(df.Time>=tmin) & (df.Time<=tmax)].mean()

def plot_Vcc(df,title):
    plt.figure()
    plt.title(title)
    plt.scatter(df.Time,1e3*df.VoltageVcc,s=1)
    plt.xlabel('$Time\ [ms]$')
    plt.ylabel('$Vcc\ [mV]$')
    plt.grid()
    plt.savefig('Figures\\'+title+'_Vcc'+'.png', dpi=300,bbox_inches='tight')

def plot_Voltage(df,title):
    plt.figure()
    plt.title(title)
    plt.scatter(df.Time,1e3*df.Voltage,s=1)
    plt.xlabel('$Time\ [ms]$')
    plt.ylabel('$Voltage\ [mV]$')
    plt.grid()
    plt.savefig('Figures\\'+title+'_Voltage'+'.png', dpi=300,bbox_inches='tight')

def plot_Current_and_Vcc(df,title):
    fig, axs = plt.subplots(2, 1)
    fig.suptitle(title)
    axs[0].scatter(df.Time,1e3*df.Current,s=1,color='green')
    axs[0].set_ylabel('$Current\ [mA]$')
    axs[0].grid(True)
    axs[1].scatter(df.Time,1e3*df.VoltageVcc,s=1)
    axs[1].set_ylabel('$Vcc\ [mV]$')
    axs[1].grid(True)
    fig.tight_layout()
    plt.xlabel('$Time\ [ms]$')
    plt.savefig('Figures\\'+title+'_Current_and_Vcc'+'.png', dpi=300,bbox_inches='tight')

def plot_fitted_curve(tsys,ysys,x_data,y_data,title):
    plt.figure()
    plt.title(title)
    plt.plot(tsys,ysys,linewidth=3,label='$Fitted\ Curve$')
    plt.plot(x_data,y_data,linewidth=1,label='$Data$')
    plt.xlabel('$Time\ [ms]$')
    plt.ylabel('$Current\ [A]$')
    plt.legend(loc='best')
    plt.grid()
    plt.savefig('Figures\\'+title+'_fitted_curve'+'.png', dpi=300,bbox_inches='tight')

def plot_Current_predicted(tsys,ysys,x,y,title):
    plt.figure()
    plt.title(title)
    plt.plot(tsys,ysys,label='$Predictions$',color='orange')
    plt.scatter(x,y,s=1,label='$Data$')
    plt.xlabel('$Time\ [s]$')
    plt.ylabel('$Current\ [A]$')
    plt.legend(loc='best')
    plt.grid()
    plt.savefig('Figures\\'+title+'_Current_predicted'+'.png', dpi=300,bbox_inches='tight')

def plot_Current_predicted_and_Vcc(tsys,ysys,t,y,u,title):
    fig, axs = plt.subplots(2, 1)
    plt.suptitle(title)
```

```python
    axs[0].plot(tsys,ysys,color='orange',label='$Predictions$')
    axs[0].scatter(t,y,s=1,label='$Data$')
    axs[0].set_ylabel('$Current_{pred}\ [A]$')
    axs[0].grid(True)
    axs[0].legend()
    axs[1].scatter(t,u,s=1)
    axs[1].set_ylabel('$Vcc\ [V]$')
    axs[1].grid(True)
    plt.xlabel('$Time\ [s]$')
    fig.tight_layout()
    plt.savefig('Figures\\'+title+'_Current_predicted_and_Vcc'+'.png', dpi=300,bbox_inches='tight'
    )

def plot_omega_predicted(tsys,ysys,title):
    plt.figure()
    plt.title(title)
    plt.plot(tsys,ysys)
    plt.xlabel('$Time\ [s]$')
    plt.ylabel('$\omega_{pred}$')
    plt.grid()
    plt.savefig('Figures\\'+title+'_omega_predicted'+'.png', dpi=300,bbox_inches='tight')

def plot_omega_predicted_and_Vcc(tsys,ysys,t,u,title):
    fig, axs = plt.subplots(2, 1)
    plt.suptitle(title)
    axs[0].plot(tsys,ysys)
    axs[0].set_ylabel('$\omega_{pred}\ [RPM]$')
    axs[0].grid(True)
    axs[1].scatter(t,u,s=1)
    axs[1].set_ylabel('$Vcc\ [V]$')
    axs[1].grid(True)
    plt.xlabel('$Time\ [s]$')
    fig.tight_layout()
    plt.savefig('Figures\\'+title+'_omega_predicted_and_Vcc'+'.png', dpi=300,bbox_inches='tight')
```

Κώδικας 3: Πρόγραμμα για σειριακή επικοινωνία του μικροελεκτή με τον υπολογιστή

```python
##############
## Script listens to serial port and writes contents into a file
##############
## requires pySerial to be installed
import serial

serial_port = '/COM5';
baud_rate = 115200; #In arduino, Serial.begin(baud_rate)
ser = serial.Serial(serial_port, baud_rate)
with open('measurements.csv', 'w+') as f:
    while True:
        line = ser.readline()
        #line = line.decode("utf-8")
        line = line.decode("latin-1")
        #line = line.decode('unicode_escape').encode('utf-8')
        print(line)
        f.write(line)
```

Κώδικας 4: Πρόγραμμα μικροελεκτή

```c
int cnt = 0;
float Voltage, VoltageVcc, Current;
int Rcontrol = 21; //Ohms
int RelayOutPin = 2 ;
unsigned long MyTime;
int sensorValue = 0;
int sensorValueVcc = 0;
const int analogInPin = A0;  // Analog input pin
const int analogInPinVcc = A5;  // Analog input pin

void setup() {
  // initialize digital pin out as an output.
  pinMode(RelayOutPin, OUTPUT);

```

```
  // turn the RELAY off
  //digitalWrite(RelayOutPin, LOW);
  digitalWrite(RelayOutPin, HIGH);

  // initialize serial communications at 9600 bps:
  //Serial.begin(9600);
  Serial.begin(115200);

  // print header
  Serial.print("Time");
  Serial.print(",");
  //Serial.print("Voltage");
  //Serial.print(",");
  //Serial.print("Current");
  //Serial.println(",");
  Serial.print("Voltage");
  Serial.print(",");
  Serial.print("VoltageVcc");
  Serial.print(",");
  Serial.print("sensorValue");
  Serial.println(",");

}

void loop() {

  if(cnt>=200){
  // turn the RELAY on
    digitalWrite(RelayOutPin, LOW);
  }

  // start time
  MyTime = millis();

  // read the analog in value:
  sensorValue = analogRead(analogInPin);

  // read the analog in value:
  sensorValueVcc = analogRead(analogInPinVcc);

  // convert the analog reading (which goes from 0 - 690) to a voltage :
  //Voltage = sensorValue * (2.93 / 690.0);
  //Current = Voltage/Rcontrol ;


   // read normal Arduino value
  Voltage = sensorValue * 5.0 / 1024.0;

   // read normal Arduino value Vcc
  VoltageVcc = sensorValueVcc * 5.0 / 1024.0;

  // read correct supply voltage
  float supply = readVcc() / 1000.0;
  float VoltageCorrected = supply / 5 * Voltage;

   // read correct supply voltage
  float VoltageCorrectedVcc = supply / 5 * VoltageVcc;

  // print the results to the Serial Monitor:
  Serial.print(MyTime);
  Serial.print(",");
  //Serial.print(Voltage,6);
  //Serial.print(",");
  //Serial.print(Current,6);
  //Serial.println(",");

  Serial.print(VoltageCorrected,6);
  Serial.print(",");
  Serial.print(VoltageCorrectedVcc,6);
  Serial.print(",");
  Serial.print(sensorValue);
```

```
  Serial.println(",");

  cnt = cnt+1;
  //delay(1);
}

long readVcc() {
  long result;
  // Read 1.1V reference against AVcc
#if defined(__AVR_ATmega32U4__) || defined(__AVR_ATmega1280__) || defined(__AVR_ATmega2560__)
  ADMUX = _BV(REFS0) | _BV(MUX4) | _BV(MUX3) | _BV(MUX2) | _BV(MUX1);
#elif defined (__AVR_ATtiny24__) || defined(__AVR_ATtiny44__) || defined(__AVR_ATtiny84__)
  ADMUX = _BV(MUX5) | _BV(MUX0);
#elif defined (__AVR_ATtiny25__) || defined(__AVR_ATtiny45__) || defined(__AVR_ATtiny85__)
  ADMUX = _BV(MUX3) | _BV(MUX2);
#else
  ADMUX = _BV(REFS0) | _BV(MUX3) | _BV(MUX2) | _BV(MUX1);
#endif
  delay(1); // Wait for Vref to settle
  ADCSRA |= _BV(ADSC); // Convert
  while (bit_is_set(ADCSRA, ADSC));
  result = ADCL;
  result |= ADCH << 8;
  result = 1126400L / result; // Calculate Vcc (in mV); 1126400 = 1.1*1024*1000
  return result;
}
```