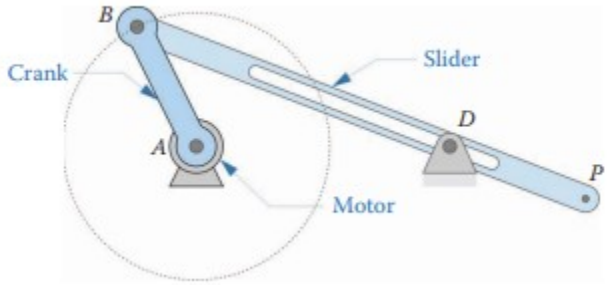
Βοηθητικό κείμενο 2: Ανάπτυξη εξισώσεων μηχανισμού και προσομοίωση στο Matlab

Για να ξεκινήσουμε τη μελέτη μας για την ανάλυση θέσης, θα χρησιμοποιήσουμε έναν από τους απλούστερους συνδέσμους που είναι ικανός για ενδιαφέρουσα κίνηση: τον τριών ράβδων ολισθητήρα-μανιβέλα. Όπως φαίνεται στην εικόνα 1, οι τρεις ράβδοι αποτελούνται από ένα στρόφαλο, ένα διωστήρα με ολισθητήρα στο κέντρο του και δύο σημεία στήριξης. Θυμηθείτε ότι το έδαφος μετράει ως ένας σύνδεσμος.

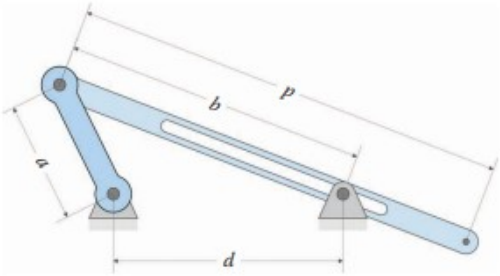
Θεωρούμε ότι ένας κινητήρας είναι στερεωμένος στον στρόφαλο έτσι ώστε να περιστρέφεται γύρω από τον πείρο A. Ο πείρος D χρησιμοποιείται για τη σύνδεση του ολισθητήρα με το έδαφος σε μια άρθρωση ημί-ολίσθησης. Ο στρόφαλος και ο ολισθητήρας στηρίζονται στο σημείο B.

Στόχος της άσκησης είναι να βρεθεί η θέση του σημείου P για οποιονδήποτε προσανατολισμό του στρόφαλου. Σημειώνεται ότι το σημείο P είναι ελεύθερο και χρησιμοποιείται απλώς για να ορίσει ένα σημείο στο τέλος του ολισθητήρα.



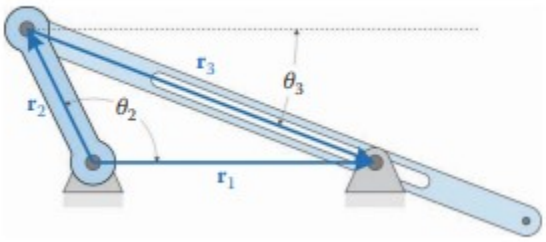
Εικόνα 1 Ο εξεταζόμενος μηχανισμός

Υπάρχουν μόνο τρεις σταθερές διαστάσεις που είναι σημαντικές για την ανάλυση θέσης, όπως φαίνεται στην εικόνα 2. Το μήκος του στρόφαλου δίνεται από το a, το συνολικό μήκος του ολισθητήρα είναι p και η απόσταση μεταξύ των στηρίξεων είναι d. Το μήκος b ορίζεται ως η απόσταση από τον πείρο του στρόφαλου A έως τον πείρο στο D. Αυτό το μήκος θα αλλάξει καθώς περιστρέφεται ο στρόφαλος και το b είναι μια από τις μεταβλητές που πρέπει να λύσουμε στην ανάλυσή μας.



Εικόνα 2 Βασικά μεγέθη του μηχανισμού

Θα ξεκινήσουμε τώρα την ανάλυση θέσης για τη σύνδεση των τριών ράβδων. Υποθέτουμε αρχικά ότι το μήκος του στρόφαλου, a, η απόσταση μεταξύ των συνδέσεων d είναι γνωστά. Επιπλέον, δεδομένου ότι οδηγούμε τον στρόφαλο χρησιμοποιώντας έναν κινητήρα, υποθέτουμε ότι η γωνία του στρόφαλου, θ_2, είναι επίσης γνωστή. Ξεκινήστε κατασκευάζοντας ένα διάγραμμα διανυσματικού βρόχου στη σύνδεση, όπως φαίνεται στην Εικόνα 3. Το διάνυσμα r_2 είναι προσαρτημένο στον στρόφαλο και έχει σταθερό μήκος a. Χρησιμοποιώντας σημειογραφία μοναδιαίου διανύσματος, μπορούμε να γράψουμε r_2 ως



Εικόνα 3 Διανυσματικός βρόγχος

$$\mathbf{r}_2 = a\mathbf{e}_2 \tag{1}$$

όπου

$$\mathbf{e}_2 = \begin{Bmatrix} \cos\theta_2 \\ \sin\theta_2 \end{Bmatrix} \tag{2}$$

είναι το μοναδιαίο διάνυσμα με διεύθυνση κατά μήκος του στρόφαλου. Το διάνυσμα r_3 είναι προσαρτημένο στον ολισθητήρα και συνδέει τον πείρο στροφάλου A με τον πείρο σύνδεσης D

$$\mathbf{r}_3 = b\mathbf{e}_3 \tag{3}$$

Το μήκος b ποικίλλει καθώς περιστρέφεται ο στρόφαλος και το μοναδιαίο διάνυσμα e_3 ευθυγραμμίζεται με τομ ολισθητήρα:

$$\mathbf{e}_3 = \left\{ \begin{array}{c} \cos\theta_3 \\ \sin\theta_3 \end{array} \right\} \tag{4}$$

Επιπλέον

$$\mathbf{r}_1 = d\mathbf{e}_1 \tag{5}$$

Εφόσον το e_1 είναι ευθυγραμμισμένο με την οριζόντια, ορίζεται ως

$$\mathbf{e}_1 = \left\{ \begin{array}{c} 1 \\ 0 \end{array} \right\} \tag{6}$$

Θα γράψουμε τώρα την εξίσωση του διανυσματικού βρόχου για τη σύνδεση των τριών ράβδων. Ένας διανυσματικός βρόχος δημιουργείται ταξιδεύοντας γύρω από τη σύνδεση, ένα διάνυσμα τη φορά, μέχρι να επιστρέψουμε στο σημείο εκκίνησης. Εφόσον τελειώνουμε στο ίδιο σημείο που ξεκινήσαμε, η συνολική απόσταση που διανύθηκε είναι μηδέν. Ξεκινώντας από το σημείο A, ο διανυσματικός βρόχος μπορεί να γραφτεί

$$\mathbf{r}_2 + \mathbf{r}_3 - \mathbf{r}_1 = 0 \tag{7}$$

Τώρα επεκτείνετε αυτήν την εξίσωση στις συνιστώσες της, χρησιμοποιώντας τους παραπάνω ορισμούς

$$a\left\{ \begin{array}{c} \cos\theta_2 \\ \sin\theta_2 \end{array} \right\} + b\left\{ \begin{array}{c} \cos\theta_3 \\ \sin\theta_3 \end{array} \right\} - d\left\{ \begin{array}{c} 1 \\ 0 \end{array} \right\} = \left\{ \begin{array}{c} 0 \\ 0 \end{array} \right\} \tag{8}$$

Ενώ η εξίσωση 8 φαίνεται να είναι μια ενιαία εξίσωση, στην πραγματικότητα αποτελείται από δύο ξεχωριστές εξισώσεις με μια συνιστώσα x και μια συνιστώσα y. Η διαίρεση αυτών των δύο εξισώσεων δίνει

$$\begin{aligned} x &: a\cos\theta_2 + b\cos\theta_3 - d = 0 \\ y &: a\sin\theta_2 + b\sin\theta_3 = 0 \end{aligned} \tag{9}$$

Το μήκος στροφάλου, a, και η απόσταση μεταξύ των στηρίξεων, d, είναι γνωστές ποσότητες, όπως και η γωνία στροφάλου θ_2. Αυτό αφήνει μόνο τη γωνία ολισθητήρα, θ_3, και την απόσταση, b, ως άγνωστες. Η μέθοδος του διανυσματικού βρόχου μας έδωσε δύο εξισώσεις, που σημαίνει ότι το πρόβλημα είναι επιλύσιμο. Ένα σωστά κατασκευασμένο διάγραμμα διανυσματικού βρόχου (ή διαγράμματα, για πιο περίπλοκες συνδέσεις) θα παρέχει πάντα τον ίδιο αριθμό εξισώσεων με τους άγνωστους. Για να λύσετε τις μεταβλητές, αναδιατάξτε πρώτα τη συνιστώσα x της εξίσωσης 9 για να λύσετε την b

$$b = \frac{d - a\cos\theta_2}{\cos\theta_3} \tag{10}$$

Εισαγάγετε αυτήν την έκφραση στο στοιχείο y της Εξίσωσης 9

$$a\sin\theta_2 + \left(\frac{d - a\cos\theta_2}{\cos\theta_3}\right)\sin\theta_3 = 0 \tag{11}$$

Και εφόσον

$$\frac{\sin\theta_3}{\cos\theta_3} = \tan\theta_3 \tag{12}$$

Μπορούμε να λύσουμε για το θ_3 ως

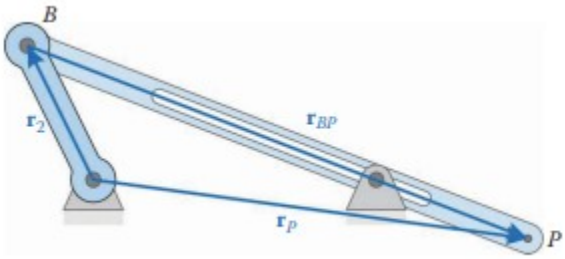
$$\tan\theta_3 = \frac{a\sin\theta_2}{a\cos\theta_2 - d} \tag{13}$$

Μόλις το θ_3 είναι γνωστό, μπορούμε να χρησιμοποιήσουμε την Εξίσωση 10 για να λύσουμε το b. Αυτό ολοκληρώνει το δύσκολο μέρος της ανάλυσης θέσης. Η δήλωση του προβλήματος, ωστόσο, μας ζητά να βρούμε τη θέση του σημείου P για οποιαδήποτε γωνία στροφάλου.

Ας ορίσουμε το διάνυσμα r_{BP}, το οποίο ξεκινά από το σημείο B και τελειώνει στο σημείο P. Επειδή αυτό το διάνυσμα έχει το ίδιο μήκος με το συνολικό μήκος του ολισθητήρα, p, και δείχνει προς την ίδια κατεύθυνση με το ρυθμιστικό, μπορούμε να γράψουμε

$$\mathbf{r}_{BP} = p\mathbf{e}_3 \tag{14}$$

Ένα διάνυσμα στο σημείο P μπορεί να βρεθεί προσθέτοντας τα διανύσματα r_2 και r_{BP}, όπως φαίνεται στην εικόνα 4



Εικόνα 4 Εύρεση θέσης σημείου P

$$\mathbf{r}_P = \mathbf{r}_2 + \mathbf{r}_{BP} \tag{15}$$

Ή, χρησιμοποιώντας τη σημειογραφία μοναδιαίου διανύσματος, έχουμε

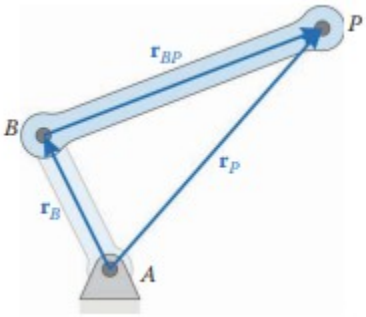
$$\mathbf{r}_P = a\mathbf{e}_2 + p\mathbf{e}_3 \tag{16}$$

Αυτή η φόρμουλα είναι αρκετά σημαντική για να απαιτεί ιδιαίτερη προσοχή. Συχνά συμβαίνει ότι γνωρίζουμε τη θέση ενός σημείου σε έναν σύνδεσμο και επιθυμούμε να γνωρίζουμε τη θέση ενός άλλου σημείου. Εξετάστε τον σύνδεσμο που φαίνεται στην Εικόνα 5. Εάν η θέση του σημείου B είναι γνωστή, τότε μπορούμε να χρησιμοποιήσουμε τον τύπο σχετικής θέσης για να βρούμε τη θέση του σημείου P.

$$\mathbf{r}_P = \mathbf{r}_B + \mathbf{r}_{BP} \tag{17}$$

Θα χρησιμοποιήσουμε τον τύπο της σχετικής θέσης αρκετά συχνά στις ενότητες που ακολουθούν και είναι αρκετά σημαντικό να ορίσουμε μια ειδική συνάρτηση MATLAB για να την υλοποιήσουμε. Αυτός ο

τύπος μπορεί επίσης να χρησιμοποιηθεί για την εξαγωγή των τύπων σχετικής ταχύτητας και σχετικής επιτάχυνσης, όπως θα δούμε.



Εικόνα 5 Υπολογισμός θέσης συνδέσμου

Χρησιμοποιώντας το Matlab για την προσομοίωση του μηχανισμού

Τώρα που έχουμε ένα σύνολο τύπων για τον υπολογισμό των γωνιών και των θέσεων των συνδέσμων στη σύνδεση των τριών ράβδων, θα βάλουμε τις γνώσεις μας στη δουλειά γράφοντας ένα πρόγραμμα MATLAB για να εκτελέσουμε τους υπολογισμούς για εμάς. Ο στόχος του προγράμματος που θα γράψουμε είναι να σχεδιάσουμε τη θέση των σημείων B και P καθώς ο στρόφαλος κάνει μια πλήρη περιστροφή. Στην πορεία, θα δημιουργήσουμε μερικές εύχρηστες συναρτήσεις MATLAB που μπορούμε να χρησιμοποιήσουμε για τη διεξαγωγή ανάλυσης θέσης πιο περίπλοκων συνδέσεων.

Ένα διάγραμμα της σύνδεσης τριών ράβδων που θα χρησιμοποιήσουμε για την ανάπτυξη του προγράμματός μας φαίνεται στην Εικόνα 6. Ο στρόφαλος έχει μήκος a = 100 mm, η απόσταση μεταξύ των στηρίξεων είναι d = 150 mm και το συνολικό μήκος του ολισθητήρα είναι p = 300 mm. Θα τοποθετήσουμε το σύστημα συντεταγμένων στο σημείο A.

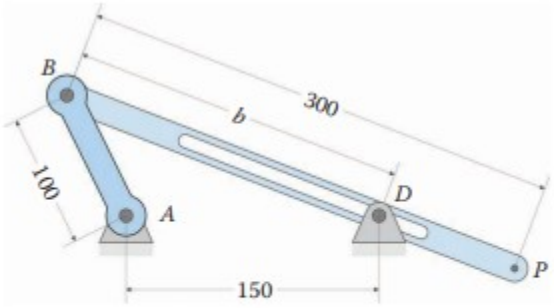
Αυτή η ενότητα είναι γραμμένη για φοιτητές που είναι νέοι στον επιστημονικό προγραμματισμό και ορισμένες από τις έννοιες θα φαίνονται αρκετά βασικές σε πιο έμπειρους προγραμματιστές. Πολλοί μαθητές φαίνεται να δυσκολεύονται να μεταφράσουν ένα σύνολο τύπων, όπως προέκυψαν στην προηγούμενη ενότητα, σε ένα πρόγραμμα για την αξιολόγηση αυτών των τύπων. Αυτή η ενότητα θα δείξει μια μέθοδο για τη συγγραφή ενός επιστημονικού προγράμματος. Φυσικά, κάθε προγραμματιστής έχει το δικό του στυλ και μπορεί να αισθάνεστε ελεύθεροι/ες να προσαρμόσετε το δικό σας πρόγραμμα όπως σας ταιριάζει (υποθέτοντας, φυσικά, ότι τα αποτελέσματα είναι τα ίδια!). Εάν δεν έχετε χρησιμοποιήσει ποτέ το MATLAB στο παρελθόν, θα πρέπει να διαβάσετε το έτερο βοηθητικό κείμενο που έχει ανέβει στο eclass.

Ένα πρόγραμμα είναι ένα σύνολο οδηγιών που δίνει ένας προγραμματιστής σε έναν υπολογιστή —όπως μια συνταγή— με στόχο την εκτέλεση μιας συγκεκριμένης εργασίας. Στην περίπτωσή μας, επιθυμούμε ο υπολογιστής να λύσει τις θέσεις των συνδέσμων και στη συνέχεια να παράγει γραφικές παραστάσεις των διαδρομών διαφόρων σημείων στη σύνδεση. Ένα από τα πρώτα πράγματα που πρέπει να παρατηρήσετε είναι ότι ορισμένες από αυτές τις εργασίες πρέπει να εκτελούνται μόνο μία φορά (π.χ. ορίζοντας τα μήκη κάθε συνδέσμου) και ορισμένες εκτελούνται πολλές φορές (π.χ. επίλυση για το μήκος του ολισθητήρα,

b και τη γωνία, θ_3, σε συγκεκριμένη γωνία στροφάλου). Θα τοποθετήσουμε τις εργασίες που θα εκτελεστούν πολλές φορές μέσα σε ένα βρόχο, αφού δεν θέλουμε να πληκτρολογούμε επανειλημμένα το σύνολο των τύπων μας. Ό,τι πρέπει να εκτελεστεί μόνο μία φορά (π.χ. καθορισμός των μηκών συνδέσμων και των εντολών σχεδίασης) θα τοποθετηθεί εκτός του βρόχου.

Όταν γράφετε ένα επιστημονικό πρόγραμμα, το πρώτο πράγμα που πρέπει να κάνετε είναι να κάνετε μια λίστα με τους στόχους του. Αυτός είναι ένας καλός τρόπος για να δοθεί μια συνολική δομή στο πρόγραμμα. Εάν κάνετε μια αρκετά λεπτομερή λίστα, ο κώδικας θα είναι σχετικά εύκολο να γραφτεί. Επιπλέον, η λίστα των στόχων μπορεί να αντιγραφεί και να επικολληθεί στο πρόγραμμα ως σχόλια που βοηθούν να εξηγηθεί ο σκοπός κάθε ενότητας του προγράμματος. Οι στόχοι του προγράμματος ανάλυσης σύνδεσης είναι:

1. Εισαγάγετε τις διαστάσεις σύνδεσης.
2. Εκτελέστε καθένα από τα παρακάτω βήματα για κάθε γωνία στροφάλου
    a. Υπολογίστε τη γωνία του ολισθητήρα, θ_3.
    b. Υπολογίστε το μήκος b μεταξύ των σημείων A και D.
    c. Χρησιμοποιήστε αυτές τις τιμές για να υπολογίσετε τις θέσεις των σημείων B και P.
3. Μόλις ολοκληρωθούν αυτοί οι υπολογισμοί, το πρόγραμμα θα πρέπει να δημιουργήσει μια γραφική παράσταση που να δείχνει τις διαδρομές των σημείων B και P καθώς ο στρόφαλος κάνει μια πλήρη περιστροφή



(All dimensions in millimeters)

Crank length: 100
Slider length: 300
Distance between ground pins: 150

Εικόνα 6 Δεδομένα του προβλήματος

Τα στοιχεία (1) και (3) πρέπει να εκτελεστούν μόνο μία φορά, ενώ οι εργασίες στο στοιχείο (2) εκτελούνται πολλές φορές—μία για κάθε γωνία στροφάλου. Επομένως, θα τοποθετήσουμε τις εργασίες στο στοιχείο (2) μέσα σε έναν βρόχο και όλες οι άλλες εργασίες θα είναι εκτός του βρόχου.

Στην κορυφή κάθε προγράμματος MATLAB θα πρέπει να πληκτρολογήσετε ένα σύνολο σχολίων που περιγράφουν τον σκοπό του προγράμματος, τον συγγραφέα του (εσείς!) και την ημερομηνία κατά την οποία γράφτηκε. Ίσως χρειαστεί να χρησιμοποιήσετε αυτό το πρόγραμμα για άλλη τάξη σε επόμενο εξάμηνο και είναι πολύ χρήσιμο να έχετε μια περιγραφή του προγράμματος στην κορυφή, ώστε να θυμάστε τι κάνει. Στην κορυφή ενός νέου σεναρίου MATLAB, πληκτρολογήστε:

```
% Threebar_Position_Analysis.m
% Conducts a position analysis on the threebar crank-slider linkage
% by Harry Valsamos, May 24, 2024
```

Σημειώστε ότι η πρώτη γραμμή δίνει το όνομα του προγράμματος: Threebar_Position_Analysis.m. Αυτό δεν είναι απαραίτητο (αφού είναι απλώς ένα σχόλιο) αλλά είναι καλή πρακτική. Θυμηθείτε ότι τα ονόματα αρχείων MATLAB δεν επιτρέπεται να έχουν κενά (ή άλλους ειδικούς χαρακτήρες) σε αυτά, επομένως χρησιμοποιήσαμε τον χαρακτήρα υπογράμμισης. Αφού πληκτρολογήσετε τα σχόλια, αποθηκεύστε το σενάριο σε μια βολική τοποθεσία χρησιμοποιώντας αυτό το όνομα αρχείου. Στις επόμενες δύο γραμμές, πληκτρολογήστε:

```
% Prepare Workspace
clear variables; close all; clc;
```

Αυτές οι γραμμές πρέπει να πληκτρολογούνται στην κορυφή όλων των scipt MATLAB. Ο σκοπός τους είναι να διαγράψουν τυχόν ορισμούς μεταβλητών από τη μνήμη, έτσι ώστε να ξεκινήσετε με μια "καθαρή πλάκα". Εάν ξεχάσετε να το κάνετε αυτό, θα διατηρήσετε όλους τους ορισμούς των μεταβλητών από την τελευταία φορά που εκτελέσατε το πρόγραμμα, μερικές φορές με πολύ εκπληκτικά και απροσδόκητα αποτελέσματα. Η εντολή close all κλείνει όλα τα παράθυρα γραφικής παράστασης που είναι ανοιχτά (όπως και πριν, με την ιδέα να ξεκινήσετε με μια καθαρή πλάκα) και η εντολή clc διαγράφει το παράθυρο εντολών (το clc σημαίνει "καθαρή γραμμή εντολών"). Στη συνέχεια, θα πρέπει να πούμε στο MATLAB τις διαστάσεις, όπως φαίνεται:

```
% Linkage dimensions
a = 0.100; % crank length (m)
d = 0.150; % length between ground pins (m)
p = 0.300; % slider length (m)
```

Αυτές οι γραμμές καθορίζουν τα μήκη κάθε συνδέσμου. Σημειώστε ότι έχουμε καθορίσει τις μονάδες για κάθε διάσταση. Αυτό είναι σημαντικό ώστε ο αναγνώστης του προγράμματός σας να γνωρίζει ποιο σύστημα μονάδων χρησιμοποιείτε.

Στη συνέχεια, θα εισαγάγουμε τις συντεταγμένες των ακίδων γείωσης, αφού δεν αλλάζουν καθώς κινείται η σύνδεση. Υπάρχουν δύο στηρίξεις, μία στο σημείο A και μία στο σημείο D

```
% Ground pins
x0 = [0;0]; % point A (the origin)
xD = [d;0]; % point D
```

Οι αγκύλες υποδεικνύουν ότι το MATLAB πρέπει να ορίσει τα x0 και xD ως διανύσματα. Κάθε διάνυσμα έχει μια συνιστώσα x και y. Ο διαχωρισμός των στοιχείων με ένα ερωτηματικό τα ορίζει ως διανύσματα στηλών με διάσταση 2 × 1. Θα χρησιμοποιήσουμε το σημείο A ως αρχή στους υπολογισμούς μας.

Κάθε μεταβλητή που αρχίζει με το γράμμα "x" θα χρησιμοποιηθεί για την αποθήκευση των συντεταγμένων ενός σημείου στη σύνδεση. Για παράδειγμα, το xB θα χρησιμοποιηθεί για την αποθήκευση των συντεταγμένων θέσης του σημείου B.

Ο καθορισμός των σταθερών στηρίξεων ήταν η τελική εργασία που επρόκειτο να εκτελεστεί μία φορά, εκτός από τις εντολές σχεδίασης, οι οποίες πρέπει να γίνουν στο τέλος του προγράμματος αφού ολοκληρωθούν όλοι οι υπολογισμοί. Τώρα είμαστε έτοιμοι να αρχίσουμε να πλαισιώνουμε τη δομή του κύριου βρόχου, ο οποίος θα εκτελέσει το σύνολο των υπολογισμών θέσης για κάθε γωνία του στροφάλου. Αρχικά, πρέπει να πάρουμε μια σημαντική απόφαση: για πόσες διαφορετικές γωνίες στροφάλου θέλουμε να υπολογίσουμε τις θέσεις των σημείων B και P; Εάν πούμε στο MATLAB να εκτελέσει τους υπολογισμούς θέσης για πολύ μικρές αυξήσεις της γωνίας του στρόφαλου (π.χ. κάθε δέκατο ή εκατοστό της μοίρας) θα παράγουμε μια πολύ «ακριβή» υπολογιστικά γραφική παράσταση με μεγάλη ακρίβεια της διαδρομής του σημείου P, αλλά με το κόστος μιας αργής εκτέλεσης. Από την άλλη πλευρά, μπορείτε να επιταχύνετε την εκτέλεση του προγράμματος εκτελώντας μόνο τους υπολογισμούς θέσης για κάθε δέκα μοίρες περιστροφής του στρόφαλου, αλλά με το κόστος μιας μη ομαλής, ανακριβούς γραφικής θέσης. Αυτό το είδος αντιστάθμισης εμφανίζεται στον προγραμματισμό αρκετά συχνά και είναι μέρος της «τέχνης» της μηχανικής.

Για αυτό το παράδειγμα, επιλέγουμε να εκτελούμε τους υπολογισμούς θέσης για κάθε 1° περιστροφής του στρόφαλου. Δεδομένου ότι θα ξεκινήσουμε με τον στροφαλοφόρο οριζόντια (στις 0°) και θα τελειώσουμε όταν ο στρόφαλος προσανατολιστεί ξανά οριζόντια (στις 360°), θα εκτελέσουμε συνολικά 361 υπολογισμούς. Δηλαδή, αν μετρήσετε από το 0 έως το 360 σε προσαυξήσεις του 1, θα έχετε συνολικά 361 υπολογισμούς θέσης. Επειδή μπορεί να θέλουμε να αυξήσουμε ή να μειώσουμε τον αριθμό των υπολογισμών στο μέλλον, θα ορίσουμε μια μεταβλητή, N, για να παρακολουθούμε αυτόν τον αριθμό.

```
N = 361; % number of times to perform position calculations
```

Στη συνέχεια, πρέπει να καθορίσουμε τον τρόπο αποθήκευσης των αποτελεσμάτων των υπολογισμών μας. Σε κάθε αύξηση της περιστροφής του στροφάλου, θα υπολογίσουμε διάφορες μεταβλητές: τη γωνία και το μήκος του ολισθητήρα (θ_3 και b) και τις συντεταγμένες των σημείων B και P στη σύνδεση. Όταν ολοκληρώσουμε τον κύριο βρόχο, θα έχουμε υπολογίσει (και αποθηκεύσει) 361 τιμές για τα θ_3 και b, καθώς και 361 x και y συντεταγμένες των σημείων B και P. Είναι πιο αποτελεσματικό να εκχωρήσουμε εκ των προτέρων μνήμη για όλα αυτά τιμές έτσι ώστε το MATLAB να μην χρειάζεται να δημιουργεί νέο χώρο αποθήκευσης σε κάθε επανάληψη του βρόχου. Αφού γίνει αυτό, το MATLAB μπορεί να τοποθετήσει νέες υπολογισμένες τιμές στον προκατανεμημένο χώρο χωρίς να χρειάζεται να βρίσκει νέο χώρο στη μνήμη κάθε φορά που περνάει από τον βρόχο. Ο απλούστερος τρόπος για προκατανομή χώρου είναι να ορίσουμε το theta2, theta3 και b ως διανύσματα μηδενικών

```
theta2 = zeros(1,N); % allocate space for crank angle
theta3 = zeros(1,N); % allocate space for slider angle
b = zeros(1,N); % allocate space for slider length
```

Αυτές οι δηλώσεις αρχικοποιούν το theta2, theta3 και b ως διανύσματα σειρών 361 μηδενικών το καθένα. Καθώς διασχίζουμε τον κύριο βρόχο, τα μηδενικά θα αντικατασταθούν από τις υπολογισμένες τιμές για κάθε μεταβλητή.

Ο καθορισμός της δομής για τις μεταβλητές θέσης είναι λίγο πιο δύσκολος. Κάθε σημείο (B και P) έχει συντεταγμένες x και y, οι οποίες πρέπει να αποθηκευτούν και οι δύο για κάθε γωνία στροφάλου. Έτσι, αντί να χρησιμοποιούμε ένα διάνυσμα μονής γραμμής όπως στις παραπάνω γωνίες, απαιτούμε δύο

σειρές για κάθε μεταβλητή θέσης. Αρχικοποιήστε τις μεταβλητές θέσης χρησιμοποιώντας τις ακόλουθες εντολές

```
[xB,xP] = deal(zeros(2,N)); % allocate space for position of B,P
```

Είναι σημαντικό να κατανοήσετε τη δομή των μεταβλητών όπως ορίζεται παραπάνω. Ο Πίνακας 1 δίνει μια γραφική αναπαράσταση της συντεταγμένης θέσης xB. Η πρώτη σειρά δίνει τις συντεταγμένες x ενώ η δεύτερη τις συντεταγμένες y. Κάθε στήλη αντιπροσωπεύει τα αποτελέσματα του υπολογισμού για μια μονή γωνία στροφάλου θ_2. Έτσι, η πρώτη στήλη περιέχει τις συντεταγμένες του σημείου B για $\theta_2 = 0°$, και η έκτη στήλη δίνει τις συντεταγμένες του σημείου B για $\theta_2 = 5°$ κ.ο.κ.

Πίνακας 1 Επεξήγηση δομής διανυσμάτων

	1	2	3	4	5	...	N
1	$xB_x(1)$	$xB_x(2)$	$xB_x(3)$	$xB_x(4)$	$xB_x(5)$	...	$xB_x(N)$
2	$xB_y(1)$	$xB_y(2)$	$xB_y(3)$	$xB_y(4)$	$xB_y(5)$	...	$xB_y(N)$

Εάν θέλουμε να πάρουμε τη συντεταγμένη y του σημείου B για τη δέκατη γωνία στροφάλου (θ2 = 9°), θα πληκτρολογήσουμε

>> xB(2,10)

στη γραμμή εντολών. Ένας από τους πιο χρήσιμους τελεστές στο MATLAB είναι η απλή άνω και κάτω τελεία. Εάν θέλουμε να έχουμε πρόσβαση σε όλες τις συντεταγμένες x του σημείου B, θα πληκτρολογήσουμε

>> xB(1,:)

στη γραμμή εντολών. Το αποτέλεσμα αυτής της δήλωσης θα ήταν ένα διάνυσμα γραμμή μήκους N που περιέχει τις συντεταγμένες x του σημείου B για κάθε γωνία στροφάλου. Ομοίως, αν θέλουμε να βρούμε τις συντεταγμένες x και y του σημείου B για την εικοστή γωνία στροφάλου θα εισάγουμε

>> xB(:,20)

στη γραμμή εντολών. Το αποτέλεσμα αυτής της δήλωσης θα ήταν ένα διάνυσμα στήλης δύο στοιχείων. το πρώτο στοιχείο θα ήταν η συντεταγμένη x του σημείου B στην εικοστή γωνία στροφάλου και το δεύτερο στοιχείο θα ήταν η συντεταγμένη y. Θα χρησιμοποιούμε τον τελεστή άνω και κάτω τελείας αρκετά συχνά στα σενάρια MATLAB μας, επομένως είναι σημαντικό να κατανοήσετε τη σύνταξή του.

Θα χρησιμοποιήσουμε έναν βρόχο for για να εκτελέσουμε τους επαναλαμβανόμενους υπολογισμούς θέσης. Οι θιασώτες του MATLAB μπορεί να συνοφρυώνονται με τη χρήση των βρόχων for για τέτοιους υπολογισμούς, αλλά μερικές φορές ο ευανάγνωστος, κατανοητός κώδικας είναι πιο σημαντικός από την καθαρή ταχύτητα εκτέλεσης! Εάν είστε γκουρού του MATLAB, μπορεί να προσπαθήσετε να διανυσματοποιήσετε τους υπολογισμούς θέσης, αλλά πιθανότατα θα αφιερώσετε περισσότερο χρόνο για τη δημιουργία προγραμμάτων από ό,τι θα εξοικονομήσετε σε ταχύτητα εκτέλεσης. Πληκτρολογήστε τα παρακάτω για βρόχο στο σενάριό σας

```
% Main Loop
for i = 1:N
```

```
end
```

Κάθε βρόχος for απαιτεί μια δήλωση τέλους και είναι καλή ιδέα να την πληκτρολογήσετε τώρα, για να μην την ξεχάσετε αργότερα. Κάθε δήλωση που τοποθετούμε μεταξύ του for και του τέλους θα εκτελείται N φορές. Η μεταβλητή i θα λάβει τις τιμές 1, 2, 3 ... N ανάλογα με την επανάληψη που εκτελούμε αυτήν τη στιγμή.

Το πρώτο πράγμα που πρέπει να κάνετε μέσα στον βρόχο είναι να προσδιορίσετε τη γωνία στροφάλου, καθώς όλοι οι επόμενοι υπολογισμοί θέσης εξαρτώνται από αυτήν. Όπως αναφέρθηκε προηγουμένως, θέλουμε να εκτελέσουμε τους υπολογισμούς με προσαυξήσεις 1°. Θυμηθείτε, ωστόσο, ότι θέλουμε να ξεκινήσουμε τους υπολογισμούς μας από 0° και να τελειώσουμε στις 360°. Η πρώτη μας εικασία για τον ορισμό της γωνίας στροφάλου, theta2, μπορεί να μοιάζει κάπως έτσι

```
for i = 1:N
    theta2(i) = i-1;



end
```

Αυτό θα κάνει το theta2 να λάβει τις τιμές 0, 1, 2, ... 360, όπως θέλετε. Ωστόσο, αυτές οι τιμές είναι σε μοίρες και οι υπολογισμοί μας πρέπει να γίνονται σε ακτίνια. Η μετατροπή μεταξύ μοιρών και ακτίνων είναι ένας πολλαπλασιαστικός παράγοντας π/180, επομένως η επόμενη εικασία μας για τον ορισμό του θήτα2 μπορεί να είναι

```
for i = 1:N
    theta2(i) = deg2rad(i-1);



end
```

Η συνάρτηση deg2rad υπάρχει σε όλες σχεδόν τις εκδόσεις του MATLAB και μετατρέπει τις μοίρες σε ακτίνια, έχοντας σαν μοναδικό όρισμα εισόδου την τιμή της γωνίας προς μετατροπή σε μοίρες.

Θα πρέπει να επιβεβαιώσετε ότι το theta2 παίρνει τιμές μεταξύ 0 και 2π αντικαθιστώντας το i = 0 και i = N στον παραπάνω τύπο. Τώρα ο στρόφαλος θα κάνει μια πλήρη περιστροφή, ανεξάρτητα από το πόσους υπολογισμούς θέσης κάνουμε. Σημειώστε την παρουσία του (i) μετά το theta2.

Χρησιμοποιούμε αυτή τη σύνταξη για να αποθηκεύσουμε κάθε γωνία στροφάλου στο διάνυσμα σειράς theta2, αντικαθιστώντας τα μηδενικά που αρχικοποιήσαμε νωρίτερα. Έτσι, στην πρώτη επανάληψη θα υπολογίσουμε το theta2(1), και στην εικοστή πέμπτη επανάληψη θα υπολογίσουμε το theta2(25). Αφού ολοκληρώσουμε τους υπολογισμούς στον κύριο βρόχο, μπορούμε να έχουμε πρόσβαση στην εικοστή γωνία στροφάλου (π.χ.) πληκτρολογώντας στη γραμμή εντολών

>> theta2(20)

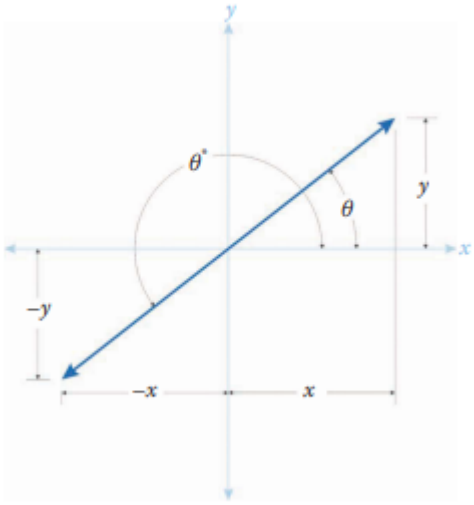
ans =

0.3316

Η μεταβλητή i είναι γνωστή ως ο δείκτης μιας συγκεκριμένης τιμής στο theta2 – σκεφτείτε τη ως τη διεύθυνση ενός συγκεκριμένου αριθμού στο διάνυσμα theta2

Τώρα που έχουμε ορίσει τη γωνία theta2, μπορούμε να αρχίσουμε να εκτελούμε τους υπολογισμούς θέσης. Πρώτα λύστε τη γωνία theta3, χρησιμοποιώντας τον τύπο 13.

Η πρώτη σας προσέγγιση μπορεί να είναι να διαιρέσετε την ποσότητα a $\sin\theta_2$ με την ποσότητα a $\cos\theta_2$ – d και να πάρετε την αντίστροφη εφαπτομένη. Αυτό θα δώσει το σωστό αποτέλεσμα εάν η γωνία θ_3 βρίσκεται εντός του πρώτου ή του τέταρτου τεταρτημορίου. Ωστόσο, όπως φαίνεται στην εικόνα 7, η συνάρτηση αντίστροφης εφαπτομένης θα δώσει το ίδιο αποτέλεσμα εάν το θ_3 βρίσκεται στο τρίτο τεταρτημόριο όπως θα δώσει εάν το θ_3 βρίσκεται στο πρώτο τεταρτημόριο, αφού



Εικόνα 7

$$\frac{y}{x} = \frac{-y}{-x}$$

Ευτυχώς, το MATLAB έχει μια ενσωματωμένη συνάρτηση, την atan2, η οποία χρησιμοποιεί ξεχωριστά ορίσματα για το y και το x, κάνοντας έτσι και τα τέσσερα τεταρτημόρια διακριτά το ένα από το άλλο.

Μετά τον υπολογισμό του θ3, βρίσκουμε εύκολα την απόσταση b χρησιμοποιώντας την Εξίσωση 10. Στο πρόγραμμα σας προσθέστε τις γραμμές:

```
for i = 1:N
    theta2(i) = deg2rad(i-1);
    theta3(i) = atan2(-a*sin(theta2(i)),d - a*cos(theta2(i)));
    b(i) = (d - a*cos(theta2(i)))/cos(theta3(i));
```

```
end
```

Εάν εκτελέσετε το πρόγραμμα τώρα, θα απογοητευτείτε όταν θα διαπιστώσετε ότι δεν συμβαίνει τίποτα! Έχουμε πει στο MATLAB να κάνει τους υπολογισμούς θέσης, αλλά να μην σχεδιάζει τίποτα. Στις απαιτήσεις που αναφέρονται παραπάνω, μας ζητήθηκε να σχεδιάσουμε τις διαδρομές των σημείων B και P, επομένως θα πρέπει να υπολογίσουμε τις θέσεις αυτών των σημείων στη συνέχεια. Εφόσον θα εκτελούμε αυτούς τους υπολογισμούς μία φορά για κάθε γωνία στροφάλου, θα πρέπει επίσης να τοποθετηθούν μέσα στον βρόχο. Ωστόσο, πριν υπολογίσουμε τις θέσεις των B και P, θα πρέπει να ορίσουμε το μοναδιαίο διάνυσμα (και το κανονικό) για κάθε σύνδεσμο. Δεν θα χρησιμοποιήσουμε όλα τα μοναδιαία διανύσματα και τα κάθετα για τους υπολογισμούς θέσης, αλλά όλα θα φανούν χρήσιμα αργότερα όταν θα πραγματοποιήσουμε ανάλυση ταχύτητας και επιτάχυνσης. Θυμηθείτε ότι οι τύποι για ένα γενικό μοναδιαίο διάνυσμα και ένα κάθετο δίνονται από

$$\mathbf{e} = \begin{Bmatrix} \cos\theta \\ \sin\theta \end{Bmatrix} \quad \mathbf{n} = \begin{Bmatrix} -\sin\theta \\ \cos\theta \end{Bmatrix}$$

Δεδομένου ότι θα υπολογίσουμε πολλά μοναδιαία διανύσματα στην ανάλυσή μας, αξίζει τον κόπο να αναπτύξουμε ένα γενικό κομμάτι κώδικα που μπορούμε να χρησιμοποιήσουμε επανειλημμένα για οποιαδήποτε σύνδεση. Ο πιο συνηθισμένος τρόπος δημιουργίας επαναχρησιμοποιήσιμου κώδικα στο MATLAB είναι η δημιουργία μιας συνάρτησης, η οποία είναι ένα ξεχωριστό αρχείο που καλείται από το κύριο πρόγραμμα όπως απαιτείται. Ένα συνηθισμένο παράδειγμα συνάρτησης είναι η ταπεινή ημιτονοειδής συνάρτηση

```
h = c*sin(delta);
```

που χρησιμοποιείται για τον υπολογισμό του ημιτόνου μιας γωνίας. Η συνάρτηση sin είναι ένα κομμάτι κώδικα που βρίσκεται βαθιά στα έγκατα του MATLAB. Ευτυχώς, δεν χρειάζεται ποτέ να μας απασχολεί η εσωτερική λειτουργία της συνάρτησης αμαρτίας, απλώς της παρέχουμε ένα όρισμα (δέλτα σε αυτήν την περίπτωση) και επιστρέφει μια απάντηση: το ημίτονο του δέλτα της γωνίας.

Ενώ πολλές γλώσσες σάς επιτρέπουν να ορίσετε συναρτήσεις μέσα στο κύριο αρχείο προγράμματος, το MATLAB θα προτιμούσε να ορίσετε τη συνάρτηση σε ξεχωριστό αρχείο στον ίδιο φάκελο με το κύριο πρόγραμμα. Η σύνταξη για μια συνάρτηση MATLAB είναι

```
function [a, b, c, …] = functionName(A, B, C, …)
```

Οι μεταβλητές A, B, C κ.λπ., είναι τιμές που περνάμε στη συνάρτηση για να κάνει τους υπολογισμούς της. Μόλις ολοκληρωθούν οι υπολογισμοί, η συνάρτηση επιστρέφει τις μεταβλητές a, b, c, κ.λπ. Πρέπει να δώσουμε στη συνάρτηση ένα όνομα (εμφανίζεται ως functionName) που να ακολουθεί τις κανονικές συμβάσεις ονομασίας αρχείων MATLAB (χωρίς κενά, πρέπει να ξεκινά με γράμμα κ.λπ. .). Θα ορίσουμε μια συνάρτηση που ονομάζεται UnitVector που θα υπολογίζει το μοναδιαίο διάνυσμα και την κανονική μονάδα, δεδομένης μιας γωνίας εισόδου θήτα. Δημιουργήστε ένα νέο σενάριο στο MATLAB και πληκτρολογήστε την παρακάτω συνάρτηση:

```
% UnitVector.m
% Calculates the unit vector and unit normal for a given angle
%
```

```
% theta = angle of unit vector
% e = unit vector in the direction of theta
% n = unit normal to the vector e
function [e,n] = UnitVector(theta)
e = [ cos(theta); sin(theta)];
n = [-sin(theta); cos(theta)];
```

Αυτή η συνάρτηση επιστρέφει ένα μοναδιαίο διάνυσμα e και ένα μοναδιαίο κάθετο n, δεδομένης της γωνίας theta. Όπως ήταν αναμενόμενο, καθένα από αυτά τα διανύσματα έχει και μια συνιστώσα x και y, και το καθένα είναι διάνυσμα στήλης με διάσταση 2 × 1. Φυσικά αυτή είναι μια πολύ, πολύ απλή συνάρτηση και οι περισσότερες συναρτήσεις του MATLAB είναι πιο περίπλοκες – σκεφτείτε το συνάρτηση atan2, για παράδειγμα. Αποθηκεύστε τη συνάρτηση ως UnitVector.m στον ίδιο φάκελο με το scipt Threebar_Position_Analysis.m. Ένα πολύ σημαντικό γεγονός σχετικά με τις συναρτήσεις είναι ότι δεν έχει σημασία πώς ονομάζετε τις μεταβλητές μέσα στη συνάρτηση, οι μεταβλητές μέσα στη συνάρτηση διαγράφονται αμέσως μόλις ολοκληρωθεί η εκτέλεση της συνάρτησης. Έχοντας εισάγει και αποθηκεύσει τη συνάρτηση UnitVector.m, εισάγετε τα ακόλουθα στο κύριο πρόγραμμα, αμέσως μετά τον υπολογισμό για το b(i).

```
for i = 1:N
    theta2(i) = deg2rad(i-1);
    theta3(i) = atan2(-a*sin(theta2(i)),d - a*cos(theta2(i)));
    b(i) = (d - a*cos(theta2(i)))/cos(theta3(i));
    % calculate unit vectors
    [e2,n2] = UnitVector(theta2(i));
    [e3,n3] = UnitVector(theta3(i));

end
```

Μπορούμε τώρα να χρησιμοποιήσουμε αυτά τα μοναδιαία διανύσματα για να υπολογίσουμε τις συντεταγμένες των σημείων B και P. Ανοίξτε ένα νέο script MATLAB και εισαγάγετε τα ακόλουθα

```
% Function FindPos.m
% calculates the position of a point on a link using the
% relative position formula
%
% x0 = position of first point on the link
% L = length of vector between first and second points
% e = unit vector between first and second points
% x = position of second point on the link
function x = FindPos(x0, L, e)
x = x0 + L * e;
```

Και πάλι, αυτή είναι μια πολύ απλή λειτουργία. Επιστρέφει τη θέση ενός σημείου σε έναν σύνδεσμο ως x δεδομένης της θέσης ενός άλλου σημείου στον σύνδεσμο, x0, καθώς και του μήκους και του μοναδιαίου διανύσματος που σχετίζονται με τον σύνδεσμο (L και e). Για το σημείο B στον στρόφαλο, το x0 θα ήταν απλώς η αρχή. Αποθηκεύστε τη συνάρτηση FindPos.m στον ίδιο φάκελο με το κύριο πρόγραμμα και εισαγάγετε τα ακόλουθα στο κύριο πρόγραμμα μετά τους υπολογισμούς μοναδιαίων διανυσμάτων μονάδων

```
for i = 1:N
    theta2(i) = deg2rad(i-1);
    theta3(i) = atan2(-a*sin(theta2(i)),d - a*cos(theta2(i)));
    b(i) = (d - a*cos(theta2(i)))/cos(theta3(i));
    % calculate unit vectors
    [e2,n2] = UnitVector(theta2(i));
    [e3,n3] = UnitVector(theta3(i));
    % solve for positions of points B and P on the linkage
    xB(:,i) = FindPos( x0,a,e2);
    xP(:,i) = FindPos(xB(:,i),p,e3);


end
```

Η σύνταξη σε αυτές τις δύο δηλώσεις μπορεί να είναι λίγο μπερδεμένη στην αρχή. Θυμηθείτε ότι υπολογίζουμε τη θέση των σημείων B και P για κάθε γωνία στροφάλου: 361 υπολογισμοί συνολικά. Εφόσον κάθε υπολογισμός για τη θέση του σημείου B καταλήγει σε δύο τιμές (τις συντεταγμένες x και y) πρέπει να χρησιμοποιήσουμε τον τελεστή άνω και κάτω τελείας για τον πρώτο δείκτη του xB. Η ποσότητα xB(:,i) αναφέρεται και στις συντεταγμένες x και y του ith υπολογισμού για τη θέση του σημείου B. Με άλλα λόγια, η άνω τελεία λέει στο MATLAB να κάνει κύκλο μέσω όλων των πιθανών τιμών αυτού του πρώτου δείκτη (στην περίπτωση αυτή, 1 και 2). Σε οποιαδήποτε επανάληψη στον βρόχο, το i λαμβάνει μία μόνο τιμή. Έτσι, η ποσότητα xB(:,i) αναφέρεται σε ένα μόνο διάνυσμα 2 × 1: οι συντεταγμένες του σημείου B στην τρέχουσα γωνία στροφάλου.

Τώρα είμαστε έτοιμοι να σχεδιάσουμε τη θέση του σημείου B. Αμέσως μετά τον βρόχο, πληκτρολογήστε την εντολή

```
plot(xB(1,:),xB(2,:))
```

Σημειώστε ότι η άνω τελεία βρίσκεται τώρα στο δεύτερο δείκτη στο xB. Η σύνταξη για την εντολή plot είναι

plot(vector of x coordinates, vector of y coordinates)

Η ποσότητα xB(1,:) δίνει ένα διάνυσμα των συντεταγμένων x για το σημείο B, ενώ η xB(2,:) δίνει όλες τις συντεταγμένες y. Όταν εκτελείτε το πρόγραμμα ανταμείβεστε από την πλοκή αυτού που φαίνεται να είναι μια έλλειψη. Εφόσον σχεδιάζουμε τη διαδρομή του σημείου B, που είναι στερεωμένο στο άκρο του στρόφαλου, η αναμενόμενη κίνηση είναι κύκλος. Θυμηθείτε ότι όλη η κίνηση σε έναν σύνδεσμο με έναν πείρο γειωμένο περιορίζεται σε κυκλικά τόξα. Αυτό που συμβαίνει είναι ότι το MATLAB κλιμακώνει τους άξονες x και y της γραφικής παράστασης για να ταιριάζει καλύτερα στο παράθυρο της γραφικής παράστασης, και ο άξονας x καταλήγει αναπόφευκτα λίγο απλωμένος. Μπορούμε να διορθώσουμε την κατάσταση πληκτρολογώντας

```
axis equal
```

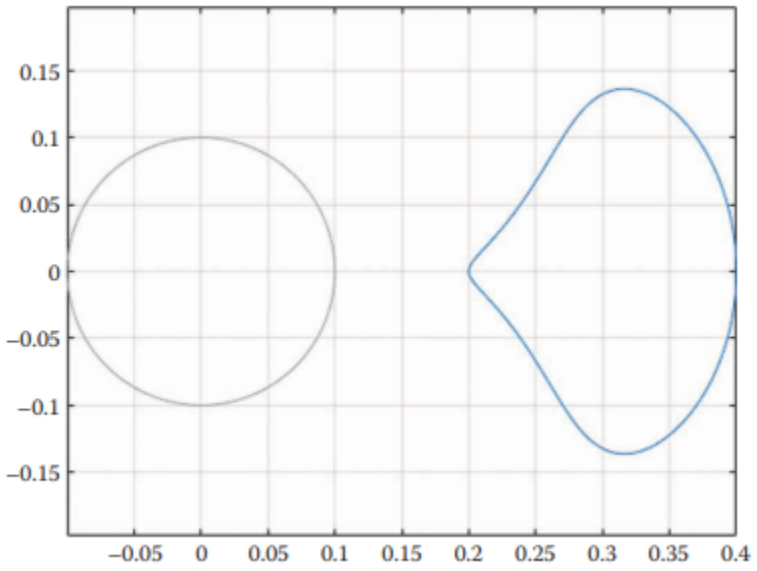
αμέσως μετά την εντολή plot. Δοκιμάστε αυτό και η πλοκή σας θα γίνει κύκλος. Δεδομένου ότι πιθανώς θέλουμε να μετρήσουμε τις συντεταγμένες των σημείων στο γράφημα, ένα πλέγμα θα ήταν επίσης χρήσιμο. Πληκτρολογιστε

```
grid on
```

μετά την εντολή άξονα για να δείτε το πλέγμα. Για να σχεδιάσετε τη θέση του σημείου P στο ίδιο σχήμα, τροποποιήστε τη δήλωση γραφικής παράστασης ως

```
plot(xB(1,:),xB(2,:))
hold on
plot(xP(1,:),xP(2,:))
axis equal
grid on
```

Όταν εκτελείτε το πρόγραμμα που περιγράφεται παραπάνω, θα πρέπει να αποκτήσετε ένα σύνολο δύο καμπυλών όπως φαίνεται στην Εικόνα 8. Έχουμε λύσει το πρόβλημα όπως δόθηκε στη δήλωση προβλήματος, αλλά υπάρχουν ακόμα μερικά πράγματα που πρέπει να προσθέσουμε για να κάνουμε την πλοκή πιο επαγγελματική. Αρχικά, θα πρέπει να προσθέσουμε έναν τίτλο και ετικέτες άξονα, ως εξής:



Εικόνα 8

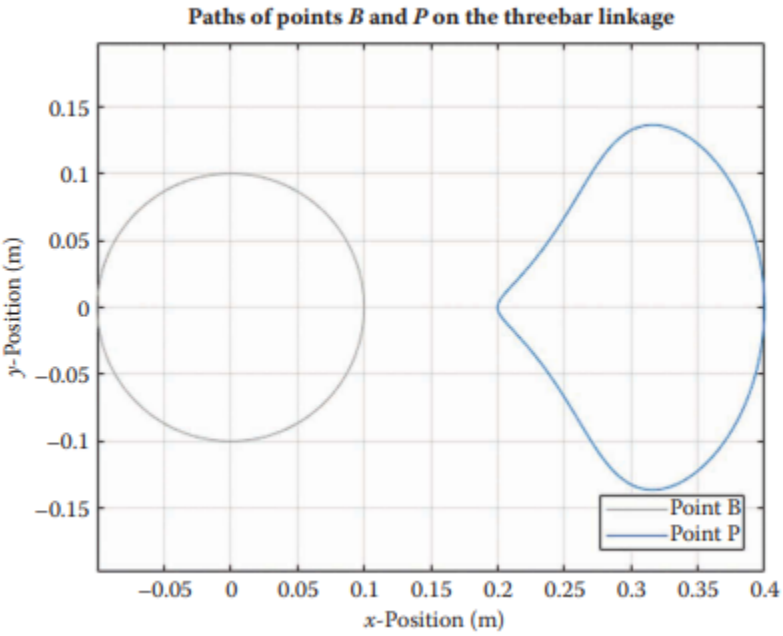
```
title('Paths of points B and P on the Threebar Linkage')
xlabel('x-position [m]')
ylabel('y-position [m]')
```

Θα πρέπει επίσης να προσθέσουμε έναν legend στην πλοκή, ώστε ο θεατής να μπορεί να διακρίνει τις δύο καμπύλες

```
legend('Point B', 'Point P','Location','SouthEast')
```

Εδώ χρησιμοποιήσαμε την ιδιότητα Location για να τοποθετήσουμε το υπόμνημα στην κάτω δεξιά γωνία του οικοπέδου. Πρέπει να είμαστε βέβαιοι ότι πληκτρολογούμε τους τίτλους των μύθων με την ίδια σειρά με την εντολή plot. Αν είχαμε πληκτρολογήσει «Σημείο P» ως πρώτο όρισμα, τα χρώματα στο

υπόμνημα δεν θα ταιριάζουν σωστά με το διάγραμμα. Μόλις τοποθετηθεί το υπόμνημα, η γραφική παράσταση έχει ολοκληρωθεί, όπως φαίνεται στην Εικόνα 9.



Εικόνα 9

Για να επικαλύψουμε μια γραφική παράσταση των συνδέσμων στα ίχνη μας, πρέπει πρώτα να πούμε στο MATLAB να συνεχίσει να σχεδιάζει στο ίδιο παράθυρο. Διαφορετικά, οποιαδήποτε νέα εντολή σχεδίου θα ανοίξει ένα νέο παράθυρο σχεδίασης. Για να το κάνετε αυτό, απλώς πληκτρολογήστε

```
hold on
```

μετά την προηγούμενη εντολή plot. Στη συνέχεια, μπορούμε να χρησιμοποιήσουμε την εντολή plot για να σχεδιάσουμε μια γραμμή για κάθε σύνδεσμο στη σύνδεση σε μια αυθαίρετη θέση. Θυμηθείτε ότι λύσαμε τις συντεταγμένες των σημείων B και P στον βρόχο και τις τοποθετήσαμε στα διανύσματα xB και xP. Ας ορίσουμε μια μεταβλητή iTheta ως το ευρετήριο του «στιγμιότυπου» που θέλουμε να σχεδιάσουμε

```
iTheta = 80;
```

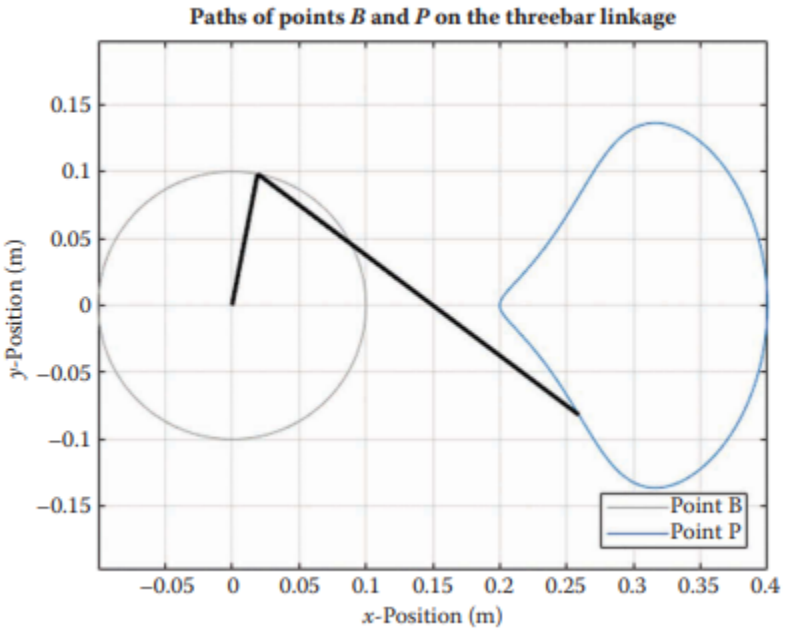
Εδώ επιλέξαμε τον υπολογισμό της 80ης θέσης για να σχεδιάσουμε. Θυμηθείτε ότι υπολογίσαμε τις θέσεις 361 φορές, οπότε το iTheta θα μπορούσε να πάρει μια τιμή μεταξύ 1 και 361. Για να σχεδιάσουμε μια γραμμή για τον στρόφαλο, θα πληκτρολογούσαμε

```
plot([x0(1) xB(1,iTheta)],...
 [x0(2) xB(2,iTheta)],'Linewidth',2,'Color','k');
```

Η εντολή plot θα είχε χυθεί στην επόμενη γραμμή, καθώς είναι μεγαλύτερη από 80 χαρακτήρες. Εδώ χρησιμοποιήσαμε τις ελλείψεις (...) για να πούμε στο MATLAB ότι η εντολή συνεχίζεται στην επόμενη γραμμή. Θυμηθείτε ότι το διάνυσμα x0 δίνει τις συντεταγμένες της αρχής (0,0). Έχουμε κάνει τη γραμμή για τη μανιβέλα πιο παχιά από τα ίχνη θέσης, ώστε να μοιάζει περισσότερο με συμπαγή σύνδεσμο. Το χρώμα είναι μαύρο, το οποίο το MATLAB συντομεύει ως «k» (για διάκριση από το μπλε, «b»). Για να σχεδιάσετε το ολισθητήρα, πληκτρολογήστε

```
hold on
plot([xB(1,iTheta) xP(1,iTheta)],...
 [xB(2,iTheta) xP(2,iTheta)],'Linewidth',2,'Color','k');
```

Εάν εκτελέσετε τον κώδικα, θα πρέπει να δείτε το διάγραμμα στην εικόνα 10. Για να κάνουμε την πλοκή ακόμα πιο εντυπωσιακή, ίσως θέλουμε να προσθέσουμε «pins» σε καθένα από τα σημεία A, B, D και P.



Εικόνα 10

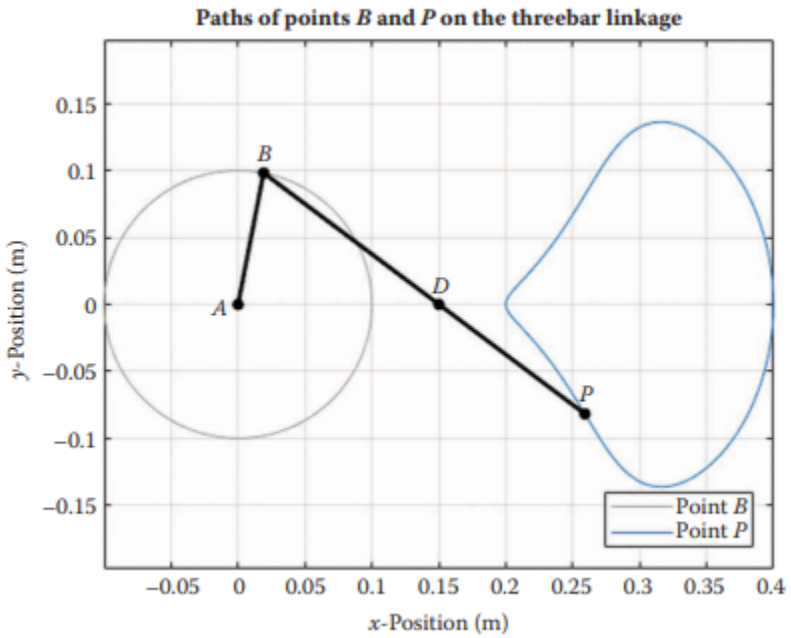
```
hold on
plot([x0(1) xD(1) xB(1,iTheta) xP(1,iTheta)],...
 [x0(2) xD(2) xB(2,iTheta) xP(2,iTheta)],...
 'o','MarkerSize',5,'MarkerFaceColor','k','Color','k');
```

Το όρισμα ‘o’ προσδιορίζει ότι πρέπει να χρησιμοποιούνται μόνο μικροί κύκλοι χωρίς γραμμές που τους συνδέουν και τα άλλα ορίσματα καθορίζουν τη διάσταση και το χρώμα των κύκλων. Ως τελικό "tweak" θα ονομάσουμε κάθε ένα από τα σημεία των οποίων τα μονοπάτια σχεδιάζουμε. Χρησιμοποιήστε τις ακόλουθες εντολές κειμένου για να τοποθετήσετε κείμενο στο οικόπεδό σας

```
% plot the labels of each pin
text( x0(1), x0(2),'A','HorizontalAlignment','center');
text(xB(1,iTheta),xB(2,iTheta),'B','HorizontalAlignment','center');
```

```
text( xD(1), xD(2),'D','HorizontalAlignment','center');
text(xP(1,iTheta),xP(2,iTheta),'P','HorizontalAlignment','center');
```

Τα δύο πρώτα ορίσματα δίνουν τις συντεταγμένες x και y του κειμένου. Το τρίτο όρισμα είναι το ίδιο το κείμενο και τα τελικά ορίσματα καθορίζουν την ευθυγράμμιση του κειμένου σε σχέση με τις συντεταγμένες που έχετε δώσει. Η τελική σας γραφική παράσταση θα πρέπει να μοιάζει με την Εικόνα 11.



Εικόνα 11

Για να αποτρέψουμε το κείμενο να επικαλύπτει τις ακίδες, καθιστώντας το πιο ευανάγνωστο, μπορούμε να προσθέσουμε μικρές μετατοπίσεις στη θέση τους (τιμή 0.015). Μια πλήρης λίστα του κώδικα ανάλυσης θέσης τριών ράβδων δίνεται παρακάτω. Βεβαιωθείτε ότι ο κώδικάς σας παράγει την ίδια γραφική παράσταση όπως φαίνεται στην εικονα 11, καθώς θα το χρησιμοποιήσουμε ως βάση για όλα τα προγράμματα σε μελλοντικά κεφάλαια.

Ο συνολικός κώδικας είναι ο ακόλουθος:

```
% Threebar_Position_Analysis.m
% Conducts a position analysis on the threebar crank-slider linkage
% by Harry Valsamos, May 24, 2024

% Prepare Workspace
clear variables; close all; clc;

% Linkage dimensions
a = 0.100; % crank length (m)
d = 0.150; % length between ground pins (m)
p = 0.300; % slider length (m)

% Ground pins
```

```
x0 = [0;0]; % point A (the origin)
xD = [d;0]; % point D

N = 361; % number of times to perform position calculations

theta2 = zeros(1,N); % allocate space for crank angle
theta3 = zeros(1,N); % allocate space for slider angle
b = zeros(1,N); % allocate space for slider length

[xB,xP] = deal(zeros(2,N)); % allocate space for position of B,P

% Main Loop
for i = 1:N
    theta2(i) = deg2rad(i-1);
    theta3(i) = atan2(-a*sin(theta2(i)),d - a*cos(theta2(i)));
    b(i) = (d - a*cos(theta2(i)))/cos(theta3(i));
    % calculate unit vectors
    [e2,n2] = UnitVector(theta2(i));
    [e3,n3] = UnitVector(theta3(i));
    % solve for positions of points B and P on the linkage
    xB(:,i) = FindPos( x0,a,e2);
    xP(:,i) = FindPos(xB(:,i),p,e3);
end

plot(xB(1,:),xB(2,:))
hold on
plot(xP(1,:),xP(2,:))
hold on
iTheta = 80;
plot([x0(1) xB(1,iTheta)],...
    [x0(2) xB(2,iTheta)],'Linewidth',2,'Color','k');
hold on
plot([xB(1,iTheta) xP(1,iTheta)],...
    [xB(2,iTheta) xP(2,iTheta)],'Linewidth',2,'Color','k');
hold on
plot([x0(1) xD(1) xB(1,iTheta) xP(1,iTheta)],...
    [x0(2) xD(2) xB(2,iTheta) xP(2,iTheta)],...
    'o','MarkerSize',5,'MarkerFaceColor','k','Color','k');
% plot the labels of each pin
text( x0(1), x0(2),'A','HorizontalAlignment','center');
text(xB(1,iTheta),xB(2,iTheta),'B','HorizontalAlignment','center');
text( xD(1), xD(2),'D','HorizontalAlignment','center');
text(xP(1,iTheta),xP(2,iTheta),'P','HorizontalAlignment','center');
axis equal
grid on
title('Paths of points B and P on the Threebar Linkage')
xlabel('x-position [m]')
ylabel('y-position [m]')

legend('Point B', 'Point P','Location','SouthEast')
```