Περιεχόμενα

Πρόλογος xi
Θέματα Προγραμματισμού xv
Λογισμικό xvii

1 Από τον Τύπο στο Πρόγραμμα 1

1.1 Απλή Αντικατάσταση Τύπου 5
Εμβαδόν Επιφάνειας Σφαίρας
MATLAB: Αριθμητικές εκφράσεις, εκχώρηση, είσοδος, έξοδος

1.2 Έλεγχος και Υπολογισμός 18
Ελάχιστο μιας Δευτεροβάθμιας σε κάποιο Διάστημα
MATLAB: Λογικές εκφράσεις, έλεγχος συνθηκών

2 Όρια και Σφάλμα 33

2.1 Κάλυψη Δίσκου 37
Άθροιση
MATLAB: Ο βρόχος `for`

2.2 Εσωτερικά/Εξωτερικά Πολύγωνα 46
Ακολουθίες
MATLAB: Ο βρόχος `while`

3 Προσέγγιση με Κλάσματα 55

3.1 Βελτιώνοντας τα 22/7 59
Κοντά στο π
MATLAB: Ένθετοι βρόχοι, μέτρηση επιδόσεων

3.2 Όχι Ακριβώς Τέλεια 69
Αριθμοί Fibonacci και Χρυσή Αναλογία
MATLAB: Πιο πολύπλοκοι βρόχοι `while`

4 Το Διακριτό έναντι του Συνεχούς 77

4.1 Ενώνοντας τα Σημεία 81
Σχεδιάζοντας Συνεχείς Συναρτήσεις
MATLAB: Διανύσματα, στοιχειώδη γραφικά

4.2 Από το Γαλάζιο στο Μοβ 97
Υπολογισμοί Χρωμάτων
MATLAB: rgb

4.3 Ένα Τρίτο Συν Ένα Τρίτο Δεν Κάνει Δύο Τρίτα 106
Το Περιβάλλον Κινητής Υποδιαστολής
MATLAB: `eps`, `inf`, `NaN`

5 Αφαίρεση 115

5.1 Μετασχηματίζοντας Ορθογώνια 119
Ένα Τετράγωνο και μια Ρίζα
MATLAB: Απλές συναρτήσεις

5.2 Το Ελλειψοειδές Οδόμετρο 132
Περίμετρος Έλλειψης
MATLAB: Συναρτήσεις με πολλαπλές παραμέτρους εισόδου

5.3 Το Πρόβλημα της Betsy Ross 144
Παράμετροι Σχεδίασης
MATLAB: Συναρτήσεις γραφικών

6 Τυχαιότητα 155

6.1 Ασφάλεια στους Αριθμούς 159
Προσομοίωση Monte Carlo
MATLAB: Περισσότερη εξάσκηση με λογικές εκφράσεις

6.2 Ζάρι και Πυξίδα 173
Τυχαίοι Περίπατοι
MATLAB: Περισσότερη εξάσκηση με βρόχους `while`

6.3 Τάξη από Χάος 180
Στάθμιση Πολυγώνων
MATLAB: Περισσότερη εξάσκηση με γραφικά και διανύσματα

7 Η Δεύτερη Διάσταση 187

7.1 Από Εδώ Έως Εκεί 191
Πίνακες Μετάβασης
MATLAB: Δισδιάστατοι πίνακες

7.2 Ισοϋψείς και Διατομές 201
Γραφική απεικόνιση της $F(x, y)$
MATLAB: Διαγράμματα ισοϋψών καμπύλων

7.3 Ψύξη 207
Προσομοίωση σε Πλέγμα
MATLAB: Ενημέρωση του `A(i,j)`

8 Αναδιάταξη 215

8.1 Κόψε και Μοίρασε 219
Το Τέλειο Ανακάτεμα
MATLAB: Περισσότερη εξάσκηση με διανύσματα και υποδείκτες

8.2 Θέση Μεγέθους 227
Ταξινόμηση
MATLAB: `sort`

9 Αναζήτηση 237

9.1 Πρότυπα σε Πρωτεΐνες 241
Γραμμική Αναζήτηση
MATLAB: Πίνακες χαρακτήρων

9.2 Τηλεφωνικός Κατάλογος Λατινικών Αριθμών 252
Δυαδική Αναζήτηση
MATLAB: Πίνακες κελιών

9.3 Αλλαγή Προσήμου 265
Διχοτομώντας για Ρίζες
MATLAB: Συναρτήσεις ως παράμετροι

10 Σημεία, Πολύγωνα και Κύκλοι 273

10.1 Πόσο Μακριά; 277
Μέτρο Απόστασης
MATLAB: Απλές δομές

10.2 Στην απομόνωση; 286
Τομή Σχημάτων
MATLAB: Πιο περίπλοκες δομές, λογικές συναρτήσεις

10.3 Όχι Τέλεια; 295
Εγγύτητα Σχήματος
MATLAB: Εξάσκηση με δομές

11 Επεξεργασία Αρχείων Κειμένου 307

11.1 Γεωγραφικό Πλάτος και Διάρκεια Ημέρας 311
Συλλογή Δεδομένων και Μετατροπή
MATLAB: Ανάγνωση δεδομένων από αρχείο κειμένου

11.2 Γειτονικά Εκατομμύρια 324
Εγγραφή και Αναπαράσταση
MATLAB: Δημιουργία αρχείων `.dat` και `.bin`

12 Πίνακες: Μέρος II 341

12.1 Ουράνιο Τόξο 345
Γραμμική Παρεμβολή και Χρωματικοί Χάρτες
MATLAB: Δημιουργία πίνακα γραμμή-ανά-γραμμή

12.2 Η Γνώση στη Γωνία 353
Διγραμμική Παρεμβολή και Σκίαση
MATLAB: Από το $f(x, y)$ στο `F(i,j)`

12.3 Επτά-επί-Πέντε 361
Ψηφιοποίηση Εικόνας
MATLAB: Πίνακες κελιών με πίνακες

12.4 Επεξεργασία Εικόνας 369
Δουλεύοντας με Αρχεία Δεδομένων Εικόνας
MATLAB: `imread`, `imwrite`, περισσότερη εξάσκηση με πίνακες

13 Επεξεργασία Ακουστικών Αρχείων 383

13.1 Το Ρολόι Χτυπάει 387
Ανάκτηση και Αναπαραγωγή
MATLAB: `wavread`, `sound`, `wavwrite`

13.2 Θόρυβος στο Τονικό 393
Συχνότητα και Δειγματοληψία
MATLAB: Περισσότερη εξάσκηση με διανύσματα

14 Διαίρει και Βασίλευε 403

14.1 Πρότυπα Μέσα σε Πρότυπα 407
Αναδρομική Επικάλυψη
MATLAB: Αναδρομικές συναρτήσεις

14.2 N και Μισό N 415
Ταξινόμηση με Συγχώνευση
MATLAB: Περισσότερη εξάσκηση με αναδρομή

14.3 Αναζητώντας Προβληματικές Περιοχές 426
Προσαρμοστική Παρεμβολή
MATLAB: Ακόμα περισσότερη εξάσκηση με αναδρομή

15 Βελτιστοποίηση 435

15.1 Η Συντομότερη Διαδρομή 439
Η Συνδυαστική Έκρηξη
MATLAB: Περισσότερη εξάσκηση με πίνακες

15.2 Το Καλύτερο Ποδήλατο 448
Περιορισμοί και Αντικειμενικές Συναρτήσεις
MATLAB: Πιο περίπλοκοι ένθετοι βρόχοι

15.3 Η Πιο Πιθανή Τροχιά 457
Μοντελοποίηση
MATLAB: Διαδραστική αναζήτηση

Παράρτημα Α. Προχωρημένα Γραφικά 467

Παράρτημα Β. Μαθηματικές Έννοιες 489

Παράρτημα Γ. MATLAB, Java και C 495

Παράρτημα Δ. Συνέντευξη Εξόδου 503

Παράρτημα Ε. Γλωσσάρι Πρώτης Χρήσης του MATLAB 507

Ευρετήριο 511

Πρόλογος

Όπως υποδηλώνει και ο τίτλος του, το βιβλίο αυτό είναι μια εισαγωγή τόσο στον προγραμματισμό με το MATLAB®, όσο και στην υπολογιστική πλευρά της επιστήμης και τεχνολογίας. Απευθύνεται κυρίως σε πρωτοετείς φοιτητές πολυτεχνικών σχολών (συμπεριλαμβανομένης της πληροφορικής) ή σχολών θετικών και φυσικών επιστημών (συμπεριλαμβανομένων των μαθηματικών). Με δεδομένες τις αναλυτικές ικανότητες αυτής της κατηγορίας φοιτητών, δε διστάζουμε να αναφερθούμε σε έννοιες της τριγωνομετρίας και της στοιχειώδους προσεγγιστικής διαδικασίας από το Λογισμό Ι. Πράγματι, είναι ενάντια στον προοδευτικό τρόπο εκπαίδευσης το να μην αναμιγνύονται ο εισαγωγικός προγραμματισμός με τα συνεχή μαθηματικά, όταν ο εκπαιδευόμενος φοιτητής είναι σε θέση να αντεπεξέλθει σε αυτό το μείγμα. Η προοδευτική εκπαίδευση έχει να κάνει με την κατανόηση των διαφορετικών τρόπων σκέψης. Γιατί να χαθεί η ευκαιρία της αντιπαράστασης του ψηφιακού τρόπου σκέψης με τον τρόπο σκέψης του συνεχούς;

Η προσέγγισή μας είναι απλή. Κάθε ενότητα ξεκινάει με την παρουσίαση ενός προβλήματος που σχετίζεται με κάποιο ευρύτερο υπολογιστικό θέμα. Η λύση συνάγεται προσεκτικά και κατά τη διάρκεια της επίλυσης του προβλήματος παρουσιάζεται οποιοδήποτε νέο στοιχείο του MATLAB είναι απαραίτητο. Αυτό ακολουθείται από μια σύντομη ενότητα "συζήτησης" που επισημαίνει κάποια πλευρά του γενικότερου θέματος που σχετίζεται με το αρχικό πρόβλημα. Η δομή αυτή ταιριάζει με την πεποίθησή μας ότι ένα εισαγωγικό μάθημα προγραμματισμού θα πρέπει να διδάσκεται μέσα από παραδείγματα. Κάθε ενότητα ολοκληρώνεται με τη δημιουργία ενός προγράμματος MATLAB και (συνήθως) μερικών συναρτήσεων. Οι ασκήσεις της κάθε ενότητας περιλαμβάνουν σχετικά απλά "Τ-προβλήματα" που βασίζονται κυρίως στον ήδη υπάρχοντα κώδικα. Τα πιο απαιτητικά "Π-προβλήματα" έχουν σχεδιαστεί για να ενισχύσουν το υπολογιστικό μήνυμα της κάθε ενότητας.

Χρησιμοποιούμε το περιβάλλον του MATLAB λόγω της φιλικότητάς του προς τον αρχάριο προγραμματιστή, αλλά και επειδή υποστηρίζει την ιδέα της ενασχόλησης με υπολογιστικές ιδέες μέσω του πειραματισμού. Αυτό είναι κάτι πολύ σημαντικό για την ανάπτυξη υπολογιστικής διαίσθησης.

> *Η ενασχόληση με προγράμματα αναπτύσσει υπολογιστική διαίσθηση.*

Η διαίσθηση είναι μια αίσθηση της κατεύθυνσης που δε διαφέρει από την αίσθηση της κατεύθυνσης που σας επιτρέπει να βρείτε το δρόμο σας, χωρίς χάρτη, στην παλιά σας γειτονιά. Το σημαντικό στοιχείο είναι ότι έχετε ξαναβρεθεί εκεί. Αν η διαίσθηση είναι μια αίσθηση της κατεύθυνσης, τότε η υπολογιστική διαίσθηση είναι μια αίσθηση της υπολογιστικής κατεύθυνσης. Αυτοί που την κατέχουν, θα είναι σε θέση να βρουν το δρόμο τους στην υπολογιστική επιστήμη και τεχνολογία του 21ου αιώνα. Η πλοήγηση απαιτεί πέντε βασικές αισθήσεις. Μέσω παραδειγμάτων και προβλημάτων προσπαθούμε να επιτύχουμε τα εξής:

1. *Ανάπτυξη όρασης για το γεωμετρικό.* Η ικανότητα οπτικοποίησης είναι πολύ σημαντική για έναν υπολογιστικό επιστήμονα. Φυσικά, τα γραφικά υπολογιστών παίζουν έναν τεράστιο ρόλο σε αυτό, αλλά τα εργαλεία οπτικοποίησης που προσφέρουν δεν αίρουν την ανάγκη συλλογισμού με γεωμετρικούς όρους. Είναι πολύ σημαντικό να είναι κανείς εξοικειωμένος με ημίτονα και συνημίτονα, με πολύγωνα και πολύεδρα, με μέτρα και εγγύτητες, κτλ.

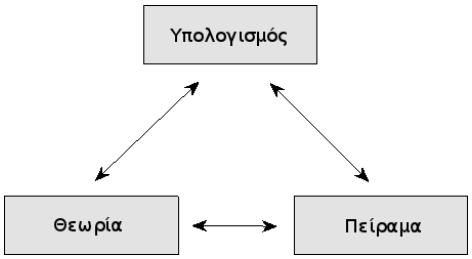
2. *Ανάπτυξη ακοής για το άκουσμα της "συνδυαστικής έκρηξης".* Πολλά προβλήματα βελτιστοποίησης και σχεδιασμού περιλαμβάνουν τεράστιους χώρους αναζήτησης με εκθετικό αριθμό πιθανοτήτων. Είναι σημαντικό να αναμένει κανείς αυτή την πολυπλοκότητα και να έχει τα μέσα να τη χειριστεί με έξυπνους ευρετικούς μηχανισμούς (heuristics).

3. *Ανάπτυξη γεύσης για το τυχαίο.* Η επιστήμη και τεχνολογία είναι γεμάτες από διαδικασίες που περιέχουν κάποιο τυχαίο παράγοντα. Το να έχει κανείς μια αίσθηση της πιθανότητας και την ικανότητα να συλλέγει και να ερμηνεύει στατιστικά δεδομένα με τον υπολογιστή, είναι ζωτικής σημασίας.

4. *Ανάπτυξη όσφρησης για τη διάσταση.* Η προσομοίωση είναι πολύ πιο υπολογιστικά απαιτητική στις τρεις διαστάσεις παρά στις δύο—η σκληρή πραγματικότητα την οποία έχουν να αντιμετωπίσουν καθημερινά οι υπολογιστικοί επιστήμονες. Για να έχει κανείς μια ακριβή ιδέα του πώς οι υπολογιστές βοηθάνε στην κατανόηση του φυσικού μας κόσμου, πρέπει πρώτα να εκτιμήσει αυτή την πραγματικότητα. Επιπλέον, το να είναι κανείς ικανός να σκέφτεται σε επίπεδο πινάκων, είναι πολύ σημαντικό για τον αποτελεσματικό, υψηλής απόδοσης υπολογισμό.

5. *Ανάπτυξη αφής για το τι είναι άπειρο, μη ακριβές και προσεγγιστικό.* Τα σφάλματα στρογγυλοποίησης ακολουθούν την αριθμητική κινητής υποδιαστολής, οι οθόνες υπολογιστών έχουν συγκεκριμένο βαθμό ανάλυσης, οι αναλυτικές παράγωγοι προσεγγίζονται με διαφορές, αντί της συνάρτησης του ημιτόνου χρησιμοποιείται ένα πολυώνυμο και τα δεδομένα που συλλέγονται σε ένα εργαστήριο έχουν ακρίβεια μόνο τριών δεκαδικών ψηφίων. Κάπως έτσι είναι η ζωή στην υπολογιστική επιστήμη και ο επιστήμονας πρέπει να είναι αρκετά αποφασιστικός ώστε να αντιμετωπίσει όλες αυτές τις αβεβαιότητες. Απαιτείται μεγάλη ικανότητα ευστάθειας για να ισορροπήσει κανείς στο όριο που χωρίζει το συνεχές από το διακριτό.

Ενώ η ανάπτυξη αυτών των πέντε αισθήσεων είναι μια σαφής προτεραιότητα, η πρωταρχική μας φιλοδοξία είναι η μετάδοση του ενθουσιασμού του υπολογισμού, μαζί με την κατανόηση των προβλημάτων του και των διασυνδέσεών του με άλλες μεθόδους. Η αλληλεπίδραση μεταξύ του υπολογισμού, της θεωρίας και του πειραματισμού, είναι ιδιαίτερα σημαντική.



Κάθε κορυφή του τριγώνου αντιπροσωπεύει ένα τύπο έρευνας και παρέχει ένα παράθυρο μέσω του οποίου μπορούμε να δούμε, σε όλο τους το εύρος, την επιστήμη και την τεχνολογία. Η ζωντάνια αυτών που βλέπουμε στο εσωτερικό του τριγώνου, εξαρτάται από τις ιδέες που ρέουν δια μέσω των πλευρών του. Μια καλή θεωρία διατυπωμένη στη γλώσσα των μαθηματικών, μπορεί να υλοποιηθεί με τη μορφή ενός προγράμματος υπολογιστή, ίσως απλά για να επιβεβαιωθεί η ορθότητά της. Η δοκιμή των αποτελεσμάτων του προγράμματος οδηγεί σε μια προσομοίωση που μπορεί να απαιτήσει τη διενέργεια ενός φυσικού πειράματος. Το πείραμα με τη σειρά του, μπορεί να αποκαλύψει μια παράμετρο που είχε παραλειφθεί στο υποκείμενο μαθηματικό μοντέλο, και ούτω καθεξής.

Υπάρχει επίσης μια ενδιαφέρουσα δυναμική και προς την αντίθετη κατεύθυνση. Ένα φυσικό πείραμα μπορεί να είναι περιορισμένο ως προς την εμβέλειά του, για οικονομικούς λόγους ή για λόγους ασφαλείας, οπότε στο επίκεντρο έρχεται η προσομοίωση στον υπολογιστή. Η πράξη της συγγραφής του προγράμματος που θα υλοποιεί την προσομοίωση, θα δώσει πιθανότατα κάποιες διευκρινίσεις, οδηγώντας έτσι σε μια νέα μαθηματική αναζήτηση. Έτσι δημιουργούνται καινοτόμα μοντέλα, που έχουν σαν αποτέλεσμα την τροποποίηση του αρχικού συνόλου των πειραμάτων, κ.ο.κ.

Η σκέψη αυτών των κρίσιμων αλληλεπιδράσεων μας θυμίζει τον μεγάλο μαθηματικό επιστήμονα Richard Hamming, ο οποίος είχε πει τη δεκαετία του 1960 ότι "ο σκοπός του υπολογισμού είναι η διαίσθηση και όχι οι αριθμοί". Προφανώς και συμφωνούμε με αυτή την άποψη. Το μήνυμα που θα πρέπει να αποκομίσει κανείς από ένα πρώτο μάθημα προγραμματισμού θα πρέπει να είναι το "διαίσθηση μέσω του υπολογισμού" και όχι απλά το "αποτελέσματα μέσω του υπολογισμού". Η επόμενη γενιά των υπολογιστικών επιστημόνων και μηχανικών θα πρέπει να σκέφτεται δημιουργικά και με ευρύτητα και ελπίζουμε το βιβλίο μας να αποτελεί μια συνεισφορά προς αυτή την κατεύθυνση.

Ευχαριστίες

Το βιβλίο αυτό προέρχεται από πολλά χρόνια εμπειρίας στη διδασκαλία του μαθήματος "Εισαγωγή στην Υπολογιστική Επιστήμη με το MATLAB" (CS 1112), στο Πανεπιστήμιο Cornell. Είμαστε ευγνώμονες σε όλους του μεταπτυχιακούς φοιτητές που διετέλεσαν βοηθοί διδασκαλίας του μαθήματος, οι οποίοι, μέσω της σκληρής τους δουλειάς, μας έδωσαν το χρόνο να βελτιώσουμε τις σημειώσεις του μαθήματος και να αναπτύξουμε ενδιαφέρουσες εργασίες. Ειδικότερα, θα θέλαμε να ευχαριστήσουμε τον κ. Tim Condon για τη συνσυγγραφή του Παραρτήματος Γ.

Γενικότερα, είμαστε τυχεροί που το ακαδημαϊκό μας σπίτι ορίζεται από δύο από τις σπουδαιότερες ακαδημαϊκές μονάδες του Πανεπιστημίου Cornell: την Πολυτεχνική Σχολή και το Τμήμα Υπολογιστικής Επιστήμης και Πληροφορικής. Η τοποθεσία είναι το παν αν θέλει κανείς να αντλεί ενέργεια τόσο από τους συναδέλφους του όσο και από τους φοιτητές του. Και εμείς έχουμε τους καλύτερους.

K.-Y. Daisy Fan
Charles F. Van Loan
Ithaca, Νέα Υόρκη

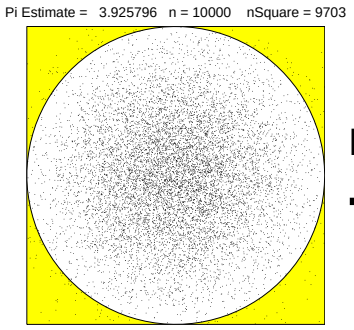
Εισαγωγικό Σημείωμα Μεταφραστών / Επιμελητών

Η *εισαγωγή στον προγραμματισμό* δε συνίσταται απλά στην εκμάθηση κάποιας γλώσσας προγραμματισμού. Αφορά κυρίως την εισαγωγή στη νοοτροπία του αλγοριθμικού τρόπου σκέψης και της υπολογιστικής επίλυσης προβλημάτων. Η *διδασκαλία του προγραμματισμού* δεν είναι η απλή εκμάθηση εντολών κάποιας συγκεκριμένης γλώσσας και μεθοδολογιών υλοποίησης προγραμμάτων. Είναι η ουσιαστική σύνδεση της αλγοριθμικής σκέψης με το προς επίλυση πρόβλημα, μέσω προσεκτικά επιλεγμένων (από διδακτικής άποψης) παραδειγμάτων διαβαθμισμένης δυσκολίας. Υπάρχουν αρκετά, καλά βιβλία για την εκμάθηση κάποιας γλώσσας προγραμματισμού αλλά τα περισσότερα από αυτά απευθύνονται σε ανθρώπους που ήδη γνωρίζουν να προγραμματίζουν σε κάποια άλλη γλώσσα. Και όσα από αυτά έχουν πιο εισαγωγικό χαρακτήρα, συχνά εστιάζουν στις λεπτομέρειες σύνταξης της γλώσσας παρά στη σύνδεσή της με την επίλυση ενδιαφερόντων προβλημάτων. Έτσι η πρώτη επαφή των φοιτητών με τον προγραμματισμό μοιάζει περισσότερο με τη (βαρετή) εκμάθηση μιας "ξένης γλώσσας" αντί να αποτελεί μια πρόκληση για την αναλυτική σκέψη τους που θα τους διαμορφώσει μια νέα οπτική και αντίληψη για τις σύγχρονες υπολογιστικές απαιτήσεις των επιστημών και της τεχνολογίας.

Η διδασκαλία της εισαγωγής στον προγραμματισμό πρέπει να έχει αρχικά ένα βασικό χαρακτηριστικό: την επιλογή μιας απλής γλώσσας προγραμματισμού που να δίνει έμφαση στον αλγοριθμικό τρόπο σκέψης και στην εφαρμογή των μεθόδων μοντέρνας διδασκαλίας προγραμματισμού που προαναφέρθηκαν. Μια ιδανική τέτοια γλώσσα προγραμματισμού είναι η γλώσσα του MATLAB. Το πρόβλημα που είχε παρατηρηθεί ήταν ότι τα υπάρχοντα βιβλία του MATLAB (αλλά και άλλων γλωσσών προγραμματισμού) ήταν περισσότερο στο επίπεδο του "εγχειριδίου χρήσης" παρά ενός διδακτικού βιβλίου που ακολουθεί τις μοντέρνες μεθόδους διδασκαλίας του προγραμματισμού. Μέχρι την εμφάνιση του πρωτοποριακού αυτού βιβλίου. Γραμμένο από έναν καταξιωμένο συγγραφέα εκπαιδευτικών βιβλίων MATLAB και υπολογιστικών μαθηματικών και μια νέα επιστήμονα με μεγάλη πείρα στη σχεδίαση και διδασκαλία εισαγωγικών και προχωρημένων μαθημάτων προγραμματισμού, έρχεται να συμπληρώσει το κενό που υπήρχε, ειδικά στην εισαγωγή στον προγραμματισμό και στην υπολογιστική επίλυση προβλημάτων.

Η δομή του βιβλίου και η όλη του φιλοσοφία αναλύεται στον πρόλογο των συγγραφέων. Εμείς, από την πλευρά των μεταφραστών / επιμελητών, προσπαθήσαμε να κρατήσουμε απαράλλαχτο όσο το δυνατόν περισσότερο το φιλικό προς τον αναγνώστη ύφος του κειμένου, με κάποιες φυσικά προσαρμογές του περιεχομένου στις ανάγκες των φοιτητών ενός Ελληνικού Πανεπιστημίου, όπου αυτό κρίθηκε απαραίτητο. Είμαστε πεπεισμένοι ότι το βιβλίο αυτό, τόσο με τη δομή και την ύλη του όσο και με τον μοντέρνο, από διδακτικής άποψης, τρόπο εισαγωγής του φοιτητή στην τέχνη του προγραμματισμού, θα καλύψει πλήρως το κενό που υπήρχε στη σωστή διδασκαλία του MATLAB και θα καταστεί χρήσιμο σε μία ευρεία γκάμα μαθημάτων, από την εισαγωγή στην πληροφορική και τον προγραμματισμό μέχρι τα υπολογιστικά μαθηματικά και τη μηχανική. Ελπίζουμε ακόμα ότι θα κεντρίσει το ενδιαφέρον των φοιτητών για τις εφαρμογές και θα τους ανοίξει ένα παράθυρο στο συναρπαστικό κόσμο της υπολογιστικής επιστήμης και τεχνολογίας.

Μιχάλης Δρακόπουλος
Ντίνος Φερεντίνος
Αθήνα, Ιούνιος 2012



Κεφάλαιο 6

Τυχαιότητα

6.1 Ασφάλεια στους Αριθμούς

Προσομοίωση Monte Carlo

6.2 Ζάρι και Πυξίδα

Τυχαίοι Περίπατοι

6.3 Τάξη από Χάος

Στάθμιση Πολυγώνων

Προκαλεί έκπληξη όταν σκεφτούμε ότι ο υπολογιστής μπορεί να χρησιμοποιηθεί για προσομοίωση τυχαίων φαινομένων. Σε τελική ανάλυση, τα προγράμματα των υπολογιστών εκτελούνται με απόλυτα προβλέψιμο τρόπο, πράγμα εντελώς αντίθετο από τη ρίψη ζαριού, την κίνηση Brown και την τυχαία μετάλλαξη. Αλλά αυτά είναι "βαθιά νερά":

> *Ο Θεός δεν παίζει ζάρια με το Σύμπαν.* (Albert Einstein)
>
> *Τα πάντα είναι τυχαία, κατεύθυνση που δε μπορείς να δεις.* (Alexander Pope)
>
> *Η τύχη βοηθάει τον προετοιμασμένο νου.* (Louis Pasteur)

Το μήνυμα εδώ είναι ότι ίσως υπάρχουνε περισσότερες διασυνδέσεις ανάμεσα στο τυχαίο και στο μη-τυχαίο από ότι φαίνεται με μια πρώτη ματιά. Αυτό ακριβώς συμβαίνει όταν χρησιμοποιούμε τις γεννήτριες ψευδοτυχαίων αριθμών `rand` και `randn` του MATLAB. Οι συναρτήσεις αυτές παράγουν στατιστικά τυχαίες ακολουθίες πραγματικών αριθμών με τις οποίες μπορούμε να προσομοιώσουμε τυχαία φαινόμενα.

Θα δείξουμε πως να υπολογίζουμε μια εκτίμηση για το π προσομοιώνοντας ένα παιχνίδι στο οποίο ρίχνουμε, με τυχαίο τρόπο, βελάκια σε ένα στόχο. Το αναμενόμενο σκορ αυτού του παιχνιδιού σχετίζεται με το εμβαδόν του κυκλικού στόχου. Αυτό είναι ένα παράδειγμα προσομοίωσης *Monte Carlo*. Στη συνέχεια θα εξετάσουμε ένα πολύ ιδιαίτερο είδος προσομοίωσης που λέγεται *τυχαίος περίπατος*. Πόσο χρειάζεται ένα ρομπότ για να φτάσει στην άκρη μιας γιγάντιας σκακιέρας όταν κινείται με τυχαίο τρόπο ένα βήμα τη φορά σε οποιαδήποτε από τις τέσσερις δυνατές κατευθύνσεις: βόρεια, νότια, ανατολικά ή δυτικά;

Θα ολοκληρώσουμε την εξόρμηση μας στον κόσμο της τύχης δείχνοντας πώς μια επαναλαμβανόμενη διαδικασία στάθμισης μπορεί “να αποκαταστήσει την τάξη”. Ένα πολύγωνο που έχει δημιουργηθεί με τυχαίο τρόπο και έχει πλευρές που διασταυρώνονται αυθαίρετα μπορεί να “ξεμπλεχτεί” μέσα από μια διαδοχή υπολογισμών στα μέσα των πλευρών. Το συγκεκριμένο πρόβλημα είναι μια μεταφορά που αναφέρεται στο γενικότερο πρόβλημα της εξομάλυνσης δεδομένων.

Προεπισκόπηση Προγραμματισμού

Έννοιες

Πιο περίπλοκες επαναλήψεις με διανύσματα, πιο περίπλοκες λογικές εκφράσεις, Monte Carlo, προσομοίωση, γεννήτρια ψευδοτυχαίων αριθμών, τυχαίος περίπατος, στάθμιση.

Χαρακτηριστικά της Γλώσσας

`rand`: Δημιουργεί ένα τυχαίο πραγματικό αριθμό από την ομοιόμορφη κατανομή στο ανοικτό διάστημα (0,1).

`randn`: Δημιουργεί ένα τυχαίο αριθμό από την τυπική κανονική κατανομή.

`seed`: Καθορίζει την αλληλουχία των παραγόμενων ψευδοτυχαίων αριθμών.

`mean`, `std`: Επιστρέφει τη μέση τιμή, τυπική απόκλιση ενός πίνακα αριθμών.

Out-of-Memory: Η μνήμη είναι περιορισμένη. Ένα ασφαλές άνω όριο για το πλήθος των στοιχείων ενός πίνακα είναι το ένα εκατομμύριο.

`bar`: Σχεδιάζει ένα ραβδόγραμμα.

`hist`: Σχεδιάζει ένα ιστόγραμμα.

`mod`: Επιστρέφει το υπόλοιπο μιας διαίρεσης.

`subplot`: Υποδιαιρεί το τρέχον παράθυρο εικόνων σε πολλαπλά σετ αξόνων.

`pause`: Σταματάει την εκτέλεση του προγράμματος για ένα αριθμό δευτερολέπτων που καθορίζεται από το χρήστη.

MatTV

Video 17. Αποσφαλμάτωση (Debugging)

Πως να χρησιμοποιήσετε τα εργαλεία αποσφαλμάτωσης του MATLAB στο Παράθυρο του Κειμενογράφου.

6.1 Ασφάλεια στους Αριθμούς

Το Πρόβλημα

Φανταστείτε ένα στόχο που σχηματίζεται από ένα τετράγωνο 2-επί-2, που έχει σαν κέντρο σκόπευσης ("bullseye") ένα μοναδιαίο δίσκο. Το γεωμετρικό κέντρο του στόχου είναι στην αρχή των αξόνων. Δείτε την Εικόνα 6.1. Ένα βελάκι εκτοξεύεται στο στόχο και καρφώνεται σε τυχαία θέση μέσα στο τετράγωνο. Λέμε ότι μια βολή είναι εύστοχη ("hit") αν το βελάκι καταλήξει μέσα στο δίσκο. Αυτό σημαίνει ότι οι συντεταγμένες (x, y) της τελικής του θέσης ικανοποιούν τη συνθήκη

$$x^2 + y^2 \leq 1.$$

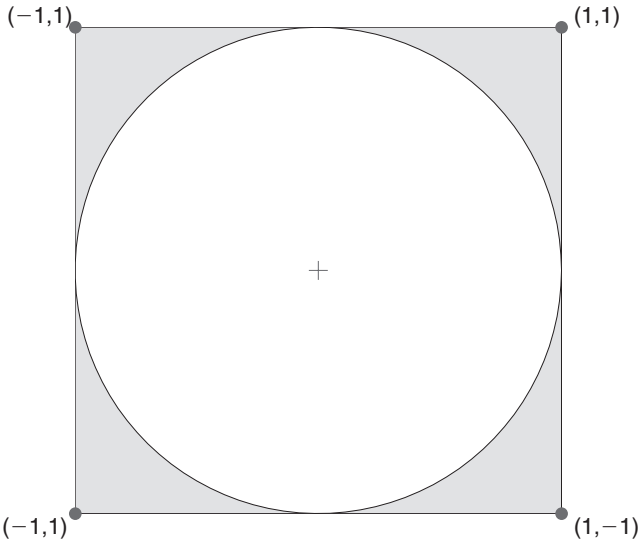
Αν ρίξουμε n βελάκια και το n είναι ένας μεγάλος αριθμός, τότε το ποσοστό των εύστοχων βολών θα πρέπει να προσεγγίζει το λόγο των δυο εμβαδών: της κυκλικής περιοχής σκόπευσης και του τετραγωνικού στόχου, δηλαδή,

$$\frac{\text{hits}}{n} \approx \frac{\pi r^2}{(2r)^2}. \tag{6.1}$$

Από αυτό προκύπτει η ακόλουθη εκτίμηση του π:

$$\pi \approx 4\,\frac{\text{hits}}{n}.$$

Γράψτε ένα πρόγραμμα που να προσομοιώνει τυχαίες βολές με 10000 βελάκια και να εμφανίζει την εκτίμηση του π που προκύπτει από αυτή τη διαδικασία. Τα βελάκια πρέπει να είναι ομοιόμορφα κατανεμημένα στην έκταση του τετραγώνου. Τι συμβαίνει αν τα βελάκια στοχεύουν το κέντρο των αξόνων; Πώς αυτό θα επηρεάσει το αποτέλεσμα του υπολογισμού του π;



Εικόνα 6.1. *Ένας Στόχος.*

Ανάπτυξη Προγράμματος

Η συνάρτηση `rand` μπορεί να χρησιμοποιηθεί για την προσομοίωση τυχαίων γεγονότων. Αν το `n` είναι ένας αρχικοποιημένος ακέραιος, τότε το script

```
for k=1:n
   r = rand(1)  % one random number
end
```

εμφανίζει μια ακολουθία αριθμών με τις ιδιότητες ότι (α) κάθε αριθμός είναι ανάμεσα στο μηδέν και το ένα και (β) η προσεκτική εξέταση των αριθμών της ακολουθίας δεν αποκαλύπτει κάποιο μοτίβο στον τρόπο παραγωγής τους, π.χ.,

```
0.95012928514718
0.23113851357429
0.60684258354179
0.48598246870930
0.89129896614890
0.76209683302739
0.45646766516834
0.01850364324822
```

Η κατανομή είναι *ομοιόμορφη* στο διάστημα ανάμεσα στο 0 και το 1 που σημαίνει ότι αν οι αριθμοί L και R ικανοποιούν τη σχέση

$$0 < L \leq R < 1,$$

τότε η πιθανότητα να είναι η τιμή του `r=rand(1)` στο διάστημα $[L, R]$ είναι $R - L$. Δηλαδή, αν εκχωρηθεί στο `n` μια μεγάλη ακέραια τιμή, τότε το script

```
count = 0;
for k=1:n
   r = rand(1);
   if L <= r && r <= R
      count = count + 1;
   end
end
p = count/n;
```

εκχωρεί στο `p` μια τιμή κοντά στο $R - L$, το μήκος του διαστήματος. Η λογική πίσω από αυτή την προσέγγιση είναι η ίδια με τη λογική πίσω από την (6.1). Αν θεωρήσουμε μια τιμή r στο επιθυμητό διάστημα $[L, R]$ σαν ένα "hit", τότε

$$\frac{\text{hits}}{n} \approx \frac{\text{μήκος επιθυμητού διαστήματος}}{\text{μήκος αρχικού διαστήματος}} = \frac{R-L}{1}.$$

Η συνάρτηση `rand` μπορεί να επιστρέψει και διανύσματα τυχαίων αριθμών. Αν το `n` έχει μια θετική ακέραια τιμή, τότε η

```
r = rand(n,1)
```

δημιουργεί ένα διάνυσμα-στήλη με n τυχαίους αριθμούς.

Για να έχουμε καλύτερη αίσθηση της *κατανομής* των τιμών, ας κατασκευάσουμε ένα ραβδόγραμμα που δείχνει το ποσοστό των τιμών της `rand` που εμπίπτουν σε κάθε δεκατημόριο. Δηλαδή, πόσες από τις τιμές που παράγει η `rand` κυμαίνονται ανάμεσα στο 0 και .1, .1 και .2, κτλ; Η συνάρτηση `rand` μας επιτρέπει να καθορίσουμε το πλήθος των τυχαίων αριθμών που θα παραχθούν. Έστω `n` ένας (μεγάλος) θετικός ακέραιος και το παρακάτω τμήμα κώδικα:[7]

```
r = rand(n,1);  % διάνυσμα n τυχαίων πραγματικών τιμών
                % ανάμεσα στο 0 και το 1
s = 10*r;       % διάνυσμα n τυχαίων πραγματικών τιμών
                % ανάμεσα στο 0 και το 10
d = ceil(s);    % διάνυσμα n τυχαίων ακεραίων επιλεγμένων
                % από το σύνολο {1,2,...,10}
```

Έχοντας δημιουργήσει το `d`, το μόνο που χρειάζεται να κάνουμε είναι να μετρήσουμε το πλήθος των τιμών σε κάθε δεκατημόριο και στη συνέχεια να "παραδώσουμε τα αποτελέσματα" στην εσωτερική συνάρτηση `bar`, που σχεδιάζει ένα ραβδόγραμμα:

```
count = zeros(10,1);
for k=1:n
   j = d(k);
   count(j) = count(j) + 1;
end
bar(count,'m')  % ραβδόγραμμα σε μοβ
```

Δείτε την Εικόνα 6.2. Αναμένουμε ότι η σχετική διαφορά ανάμεσα στα ύψη των ράβδων θα μειώνεται καθώς αυξάνεται το n.

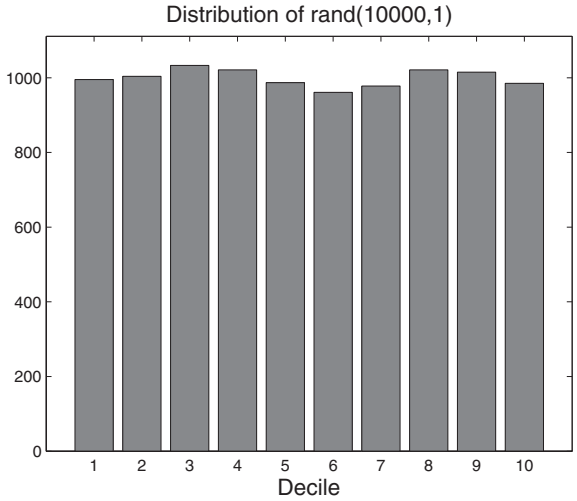
Δεν σκοπεύουμε να εμβαθύνουμε στη στατιστική ανάλυση, αλλά με τη `rand` στη διάθεσή μας μπορούμε να έχουμε μια διαισθητική αντίληψη για τη στατιστική διακύμανση. Κάτω από αυτό το πρίσμα, οι δυο πιο σημαντικές παράμετροι που περιγράφουν ένα σύνολο δεδομένων είναι η *μέση τιμή* και η *τυπική απόκλιση*. Αυτά τα στατιστικά στοιχεία, μετράνε αντίστοιχα το μέσο όρο και τη διασπορά των δεδομένων και υπολογίζονται με ευκολία χρησιμοποιώντας τις εσωτερικές συναρτήσεις `mean` και `std` του MATLAB. Για παράδειγμα, το επόμενο τμήμα κώδικα εμφανίζει τη μέση τιμή και την τυπική απόκλιση ενός μεγάλου διανύσματος τα στοιχεία του οποίου ακολουθούν την ομοιόμορφη κατανομή:[8]

```
n = 1000000;
r = rand(n,1);
mu = mean(r)
sigma = std(r)
```

Όσο αυξάνεται το n, οι τιμές των `mu` και `sigma` προσεγγίζουν τα $\mu = 0.5000$ και $\sigma = 1/\sqrt{12}$, δηλαδή τη μέση τιμή και την τυπική απόκλιση της ομοιόμορφης κατανομής.

[7] Το μηδέν δεν συμπεριλαμβάνεται στο σύνολο των πιθανών ακέραιων τιμών για το `d` γιατί η `rand` παράγει τυχαίες τιμές στο *ανοικτό* διάστημα (0,1).

[8] Κάθε φορά που αναφερόμαστε στην "ομοιόμορφη κατανομή" εννοούμε την ομοιόμορφη κατανομή στο διάστημα (0,1).



Εικόνα 6.2. *Δειγματοληψία της Ομοιόμορφης Κατανομής με τη* `rand`.

Ας επιστρέψουμε στο πρόβλημα υπολογισμού του π ρίχνοντας βελάκια, που θέσαμε στην αρχή της ενότητας. Μπορούμε να χρησιμοποιήσουμε τη `rand` για να παράγουμε τυχαίες τιμές σε ένα προκαθορισμένο διάστημα όπως το $(-1, 1)$ διευρύνοντας και μεταθέτοντας το $(0, 1)$. Όπως και με τη δημιουργία των δεδομένων για την Εικόνα 6.2, έχουμε

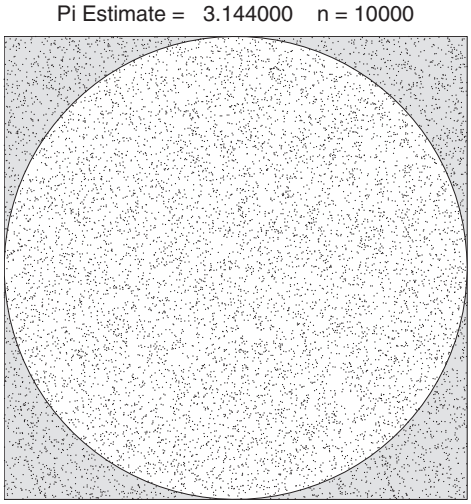
```
r = rand(n,1); % διάνυσμα n τυχαίων πραγματικών τιμών
               % ανάμεσα στο 0 και το 1
s = 2*r;       % διάνυσμα n τυχαίων πραγματικών τιμών
               % ανάμεσα στο 0 και το 2
t = s-1;       % διάνυσμα n τυχαίων πραγματικών τιμών
               % ανάμεσα στο -1 και το 1
```

Για παράδειγμα,

$$\texttt{r} = \begin{bmatrix} 0.8147 \\ 0.9058 \\ 0.1270 \\ 0.9134 \\ 0.6324 \end{bmatrix} \quad \Rightarrow \quad \texttt{s} = \begin{bmatrix} 1.6294 \\ 1.8116 \\ 0.2540 \\ 1.8268 \\ 1.2647 \end{bmatrix} \quad \Rightarrow \quad \texttt{t} = \begin{bmatrix} 0.6294 \\ 0.8116 \\ -0.7460 \\ 0.8268 \\ 0.2647 \end{bmatrix}.$$

Έτσι, το παρακάτω τμήμα κώδικα

```
hits = 0; x = -1+2*rand(n,1); y = -1+2*rand(n,1);
for k=1:n
   if x(k)^2 + y(k)^2 <= 1
      hits = hits + 1;
   end
end
piEstU = 4*(hits/n);
```



Εικόνα 6.3. *Εκτίμηση του π με Βελάκια που Ρίχνονται Ομοιόμορφα.*

προσομοιώνει ρίψεις με n βελάκια, μετράει τον αριθμό των εύστοχων βολών ("hits"), και υπολογίζει μια εκτίμηση για το π. Ένα δείγμα του αποτελέσματος εμφανίζεται στην Εικόνα 6.3.

Η μεθοδολογία οργάνωσης μια τυχαίας προσομοίωσης (παιχνίδι) της οποίας το αποτέλεσμα (σκορ) λέει κάτι για το πρόβλημα που αντιμετωπίζουμε ονομάζεται *Monte Carlo*. Προσεγγίσαμε το πρόβλημα υπολογισμού του π προσομοιώνοντας ένα παιχνίδι με βελάκια όπου το σκορ $4(\text{hits}/n)$ μας διαφωτίζει σχετικά με την τιμή του.

Τι συμβαίνει αν "σημαδέψουμε" με τα βελάκια το κέντρο του στόχου; Αυτό πιθανότατα θα οδηγήσει σε μεγαλύτερη αναλογία εύστοχων βολών ("hits") και σε μια "φουσκωμένη" εκτίμηση του π. Για να διερευνήσουμε αυτό το "σενάριο" με μεγαλύτερη ακρίβεια, τροποποιούμε την παραπάνω προσομοίωση με τα βελάκια έτσι ώστε να βασίζεται τώρα στην *κανονική κατανομή*. Αυτή είναι η γνωστή κατανομή σε σχήμα καμπάνας που εμφανίζεται στην Εικόνα 6.4. Η εκχώρηση

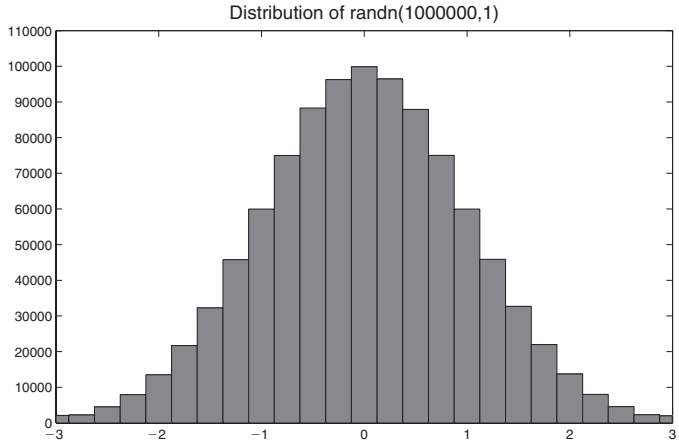
```
r = randn(n,1);
```

δημιουργεί ένα διάνυσμα με τιμές από την κανονική κατανομή, με μέση τιμή μηδέν και τυπική απόκλιση ένα. Το ιστόγραμμα στην Εικόνα 6.4 παράγεται χρησιμοποιώντας τη `randn` και τη `hist`, που είναι μια εσωτερική συνάρτηση που σχεδιάζει ιστογράμματα:

```
r = randn(1000000,1); % Ένα εκατομμύριο δείγματα
x = linspace(-3,3,25);% Κέντρα 25 διαστημάτων στο [-3,3]
hist(r,x)             % Αριθμός δειγμάτων ανά διάστημα
```

Η συντριπτική πλειονότητα (για την ακρίβεια πάνω από 98%) των τιμών r βρίσκονται στο διάστημα $[-3, +3]$. Ωστόσο, με την κανονική κατανομή υπάρχει πάντα η πιθανότητα να προκύψει μια αφύσικα μεγάλη ή μικρή τιμή.

Για δειγματοληψία από μια κανονική κατανομή με μέση τιμή `mu` και τυπική απόκλιση `sigma`, απλά κλιμακώνουμε και μετατοπίζουμε κατάλληλα τις τιμές·



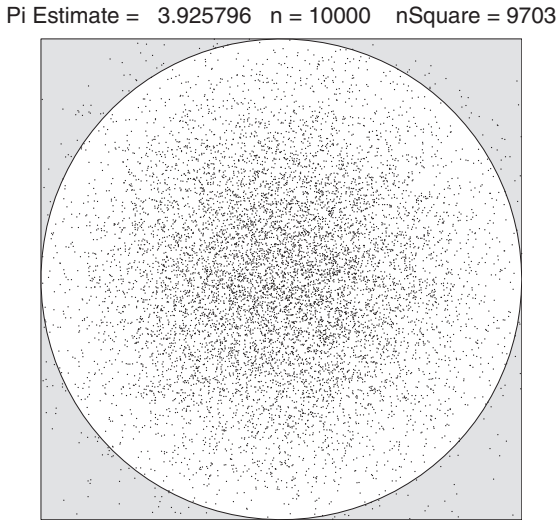
Εικόνα 6.4. *Η Κανονική Κατανομή.*

```
r = mu + sigma*randn(n,1);
```

Έτσι, μπορούμε να προσομοιώσουμε τη στόχευση με τυχαία βελάκια προς το κέντρο του στόχου (το `mu` είναι μηδέν) δημιουργώντας τις συντεταγμένες (x, y) των βολών ως εξής:

```
x = sigma*randn(n,1);
y = sigma*randn(n,1);
```

Δείτε στην Εικόνα 6.5 το στόχο μετά από ρίψεις με τυπική απόκλιση `sigma` ίση με 0.4. Για να εκτιμήσουμε το π, θα μπορούσαμε να προχωρήσουμε όπως και με τις ομοιόμορφες



Εικόνα 6.5. *Εκτίμηση του π με Στοχευμένα Βελάκια.*

ρίψεις και απλώς να καταγράψουμε τις βολές που κατέληξαν μέσα στον κύκλο. Ωστόσο, επειδή μερικές βολές δε θα καταλήξουν καν μέσα στο τετράγωνο, είναι λογικότερο να προσεγγίσουμε το π με το $(4/n_{square})\cdot$hits αντί με το $(4/n)\cdot$hits, όπου n_{square} είναι ο αριθμός από τα βελάκια που θα πέσουν μέσα στο τετράγωνο. Το τελικό script `Eg6_1`, που δίνεται παρακάτω, επιλύει το πρόβλημα εκτίμησης του π με αυτό το πηλίκο.

Το script περιλαμβάνει ένα σημαντικό στοιχείο που έχει να κάνει με την επαναληψιμότητα των πειραμάτων. Χωρίς τις εντολές `rand('seed',0)` και `randn('seed',0)`, θα παίρναμε διαφορετικά αποτελέσματα κάθε φορά που εκτελείται το script. Η δημιουργία τυχαίων αριθμών ξεκινάει με μια τιμή αρχικοποίησης της γεννήτριας που ονομάζεται *seed*. Σε γενικές γραμμές, το MATLAB αναλαμβάνει αυτή την αρχικοποίηση εκτός εάν παρέμβει ο χρήστης δίνοντας τιμή στο seed. Με το *ίδιο* seed, θα παράγεται η *ίδια* ακολουθία τυχαίων αριθμών κάθε φορά που εκτελείται η προσομοίωση. Η προσομοίωση επιβεβαιώνει το προφανές: αν σημαδέψουμε με τα βελάκια καταλήγουμε σε μια "φουσκωμένη" εκτίμηση του π.

Ολοκληρώνουμε αυτή την ενότητα εισάγοντας τεχνικές που μπορούν να χρησιμοποιηθούν για να απλοποιήσουν και να εκφράσουν σε διανυσματική μορφή δομημένες αλληλουχίες λογικών πράξεων όπως αυτές που προκύπτουν στο script `Eg6_1`. Ξεκινάμε με την ιδέα μια μεταβλητής λογικού τύπου, που παίρνει δηλαδή λογικές (boolean) τιμές. Μια λογική έκφραση όπως η `x < y` είναι είτε αληθής ή ψευδής. Το ένα και το μηδέν χρησιμοποιούνται για να παρασταθούν αυτές οι λογικές τιμές οι οποίες μπορούν να αποθηκευτούν για να χρησιμοποιηθούν και στη συνέχεια. Έτσι, αν οι `x` και `y` είναι αρχικοποιημένες βαθμωτές μεταβλητές με πραγματικές τιμές, τότε η

```
B = x < y;
```

εκχωρεί στο `B` την τιμή 1 (αληθής) αν η τιμή του `x` είναι μικρότερη από την τιμή του `y` και την τιμή 0 (ψευδής) στην αντίθετη περίπτωση. Το `B` είναι μια *λογική μεταβλητή*. Σε περίπλοκες καταστάσεις, η αναγνωσιμότητα συχνά ενισχύεται με την αποθήκευση ενδιάμεσων λογικών τιμών. Να ένα τμήμα κώδικα που ελέγχει αν ο ακέραιος `y` είναι ένα έγκυρο δίσεκτο έτος:

```
DivBy4      = (mod(y,4)==0);
NotDivBy100 = (mod(y,100)~=0);
DivBy400    = (mod(y,400)==0);
if (DivBy4 && NotDivBy100) || DivBy400
   disp('Δίσεκτο Έτος')
else
   disp('Όχι Δίσεκτο Έτος')
end
```

Παρατηρήστε πώς η εντολή `if-else` "διαβάζεται" όπως ακριβώς η φυσική μας γλώσσα περιγράφει τον κανόνα για ένα δίσεκτο έτος. Οι εκφράσεις `mod` περικλείονται σε παρενθέσεις για την "άνεση του αναγνώστη". Η εκχώρηση

```
DivBy4 = mod(y,4)==0;
```

είναι επίσης σωστή αλλά "χτυπάει στο μάτι".

Το Script `Eg6_1`

```
% Script Eg6_1
% Υπολογισμός του Pi με ομοιόμορφη και με κανονική κατανομή

% Αρχικοποιήσεις
n = 10000; rand('seed',0); randn('seed',0);

% Ρίξε τα Βελάκια Ομοιόμορφα...
x = -1+2*rand(n,1);
y = -1+2*rand(n,1);
hits = 0;
for k=1:n
    % Έλεγξε τη βολή για το βελάκι k...
    if x(k)^2 + y(k)^2 <= 1
        hits = hits + 1;
    end
end
piEstU = 4*(hits/n);

% Ρίξε τα Βελάκια Σημαδεύοντας...
sigma = .4;
x = sigma*randn(n,1);
y = sigma*randn(n,1);
hits = 0;
nSquare = 0;
for k=1:N
    % Έλεγξε τη βολή για το βελάκι k...
    if abs(x(k))<=1 && abs(y(k))<=1
        nSquare = nSquare + 1;
        if x(k)^2 + y(k)^2 <= 1
            hits = hits + 1;
        end
    end
end
piEstN = 4*(hits/nSquare);

% Εμφάνισε τις εκτιμήσεις...
clc
fprintf('n: %1d\n',n)
fprintf('Εκτίμηση του Pi με Ομοιόμορφη Κατανομή: 7.5f\n',...
         piEstU)
fprintf('Εκτίμηση του Pi με N(0,%5.2f)   : %7.5f\n',...
         sigma,piEstN)
```

Δείγμα Εξόδου του Script `Eg6_1`

```
n: 10000
Εκτίμηση του Pi με Ομοιόμορφη Κατανομή: 3.14400
Εκτίμηση του Pi με N(0, 0.40)         : 3.92580
```

Το MATLAB υποστηρίζει λογικές πράξεις ανάμεσα σε διανύσματα με το ίδιο μήκος και προσανατολισμό[9]. Έτσι, με τις εντολές

```
x = [10 20 50 40 80];
y = [7 25 60 30 90];
B = (x < y);
```

εκχωρείται στο `B` το διάνυσμα-γραμμή `[0 1 1 0 1]` γιατί

`x(1) < y(1)`	είναι ψευδής
`x(2) < y(2)`	είναι αληθής
`x(3) < y(3)`	είναι αληθής
`x(4) < y(4)`	είναι ψευδής
`x(5) < y(5)`	είναι αληθής.

Παρατηρήστε ότι η τιμή του `sum(B)` είναι 3, το πλήθος των αληθών τιμών που σχετίζεται με τη σύγκριση των δυο διανυσμάτων.

Χρησιμοποιώντας αυτές τις ιδέες μπορούμε να απλοποιήσουμε τη διαχείριση της προσομοίωσης ρίψεων τα βελάκια στο script `Eg6_1`. Αν το `n` είναι ένας αρχικοποιημένος ακέραιος, τότε με τις εντολές

```
x = -1 + 2*rand(n,1);
y = -1 + 2*rand(n,1);
S = (x.^2 + y.^2 <= ones(n,1));
hits = sum(S);
```

εκχωρείται στο `hits` το πλήθος των ρίψεων που κατέληξαν μέσα στο μοναδιαίο κύκλο. Το διάνυσμα λογικών τιμών `S` είναι ένα διάνυσμα 0-1 με την ιδιότητα ότι το `B(i)` είναι 1 αν το βελάκι i είναι ένα "hit" και 0 αν δεν είναι.

Συζήτηση: Δημιουργία Ψευδοτυχαίων Αριθμών

Με την επιστήμη και την τεχνολογία να βασίζονται σε τόσο μεγάλο βαθμό στην υπολογιστική προσομοίωση και με τόσες πολλές προσομοιώσεις να μιμούνται τυχαία γεγονότα, πολύ σημαντικό οι τεχνικές που χρησιμοποιούνται για γεννήτριες τυχαίων αριθμών να είναι στατιστικά ορθές. Ενώ η σημερινή κατάσταση των τεχνικών αυτών είναι εξαιρετική, είναι σημαντικό να θυμόμαστε μερικές από τις αποτυχίες του παρελθόντος μόνο και μόνο για να εκτιμήσουμε πόσο υψηλό είναι το διακύβευμα.

Για να δούμε τι μπορεί να πάει στραβά όταν προσπαθούμε να δημιουργήσουμε μια ακολουθία τυχαίων αριθμών, θεωρούμε τη μέθοδο *middle squaring*. Όταν εφαρμόζεται σε

[9] Σ.τ.Μ. Να είναι και τα δύο διανύσματα-γραμμή ή διανύσματα-στήλη

4-ψήφιους ακέραιους, δημιουργεί την ακολουθία $\{x_1,\ x_2,\ \ldots\}$ σύμφωνα με την ακόλουθη "συνταγή"

```
x(k+1) = rem(floor(x(k)^2/100),10000).
```

Με άλλα λόγια, για να πάρουμε τον επόμενο τυχαίο ακέραιο υψώνουμε στο τετράγωνο τον τρέχοντα τυχαίο ακέραιο και κρατάμε τα τέσσερα ενδιάμεσα ψηφία από το 8-ψήφιο αποτέλεσμα. Μερικές δοκιμές μας πείθουν ότι δεν υπάρχει κάποιο ευδιάκριτο μοτίβο στον τρόπο δημιουργίας των αριθμών αυτών. Ωστόσο, μια πιο επιστημονική αξιολόγηση της τεχνικής αποκαλύπτει την παρουσία περιοδικότητας με μικρούς κύκλους επανάληψης, π.χ., $2100 \to 4100 \to 8100 \to 6100 \to 2100$. Αυτό υπονομεύει πλήρως την ιδέα ότι τα x_k δεν συσχετίζονται.

Παρ' όλο που η μέθοδος middle squaring δεν κατέχει κάποια σημαντική θέση στην ιστορία των γεννητριών τυχαίων αριθμών, έχουν υπάρξει γεννήτριες ευρείας χρήσης που έχουν ανησυχητικά μικρούς κύκλους επανάληψης όπως η παραπάνω. Ευτυχώς, είναι δυνατόν να ελέγξουμε στατιστικά την αξιοπιστία των σύγχρονων γεννητριών τυχαίων αριθμών όπως οι `rand` και `randn`. Αυτό είναι σημαντικό επειδή, σε τελική ανάλυση, οι ακολουθίες που παράγουν αυτές οι συναρτήσεις είναι ντετερμινιστικές, δηλαδή, είναι απόλυτα προβλέψιμες εφόσον γνωρίζουμε τον αλγόριθμό παραγωγής τους. Γι' αυτό, από τεχνική άποψη, οι `rand` και `randn` είναι γεννήτριες *ψευδοτυχαίων* αριθμών.

Επισκόπηση MATLAB

rand

Αν το `n` έχει αρχικοποιηθεί σε ένα θετικό ακέραιο, τότε η `rand(n,1)` είναι ένα διάνυσμα-στήλη n στοιχείων με τιμές που ακολουθούν την ομοιόμορφη κατανομή στο *ανοιχτό* διάστημα (0,1). Για να πάρουμε ένα διάνυσμα-γραμμή, χρησιμοποιούμε τη `rand(1,n)`. Σημειώστε ότι τα ορίσματα της `rand` καθορίζουν *πόσες τιμές* θα παραχθούν, και όχι το διάστημα στο οποίο κυμαίνονται. Αυτό το διάστημα είναι πάντα το (0,1).

randn

Αν το `n` έχει αρχικοποιηθεί σε ένα θετικό ακέραιο, τότε η `randv(n,1)` είναι ένα διάνυσμα-στήλη n στοιχείων με τιμές που ακολουθούν την κανονική κατανομή με μέση τιμή μηδέν και τυπική απόκλιση ένα. Για να πάρουμε ένα διάνυσμα-γραμμή, χρησιμοποιούμε τη `randn(1,n)`.

seed

Βάζοντας τις εντολές `rand('seed',0)` και `randn('seed',0)` στην αρχή ενός script μας εξασφαλίζει ότι το script θα μας δώσει τα ίδια αποτελέσματα κάθε φορά που το τρέχουμε.

mean και std

Αν το `x` είναι ένα αρχικοποιημένο διάνυσμα, τότε ο `mean(x)` είναι ο μέσος όρος των τιμών και η `std(x)` είναι η τυπική απόκλιση.

sum

Αν το `x` είναι ένα διάνυσμα, τότε η τιμή του `sum(x)` είναι το άθροισμα των στοιχείων του.

max και min

Αν το `x` είναι ένα διάνυσμα, τότε η τιμή του `max(x)` είναι το μέγιστο στοιχείο του `x`. (Σημειώστε ότι το `max(abs(x))` είναι το μέγιστο κατά απόλυτη τιμή στοιχείο του `x`). Μια αναφορά της μορφής `[alfa,i] = max(x)` εκχωρεί στο `alfa` την τιμή του μέγιστου στοιχείου του `x` και στο `i` τη θέση αυτής της τιμής. Ανάλογη είναι και η συνάρτηση `min` για το ελάχιστο.

bar

Αν το `x` είναι ένα αρχικοποιημένο διάνυσμα, τότε η `bar(x)` εμφανίζει τις τιμές του σε μορφή ραβδογράμματος. Ο αριθμός των ράβδων είναι το μήκος του `x`.

hist

Αν τα `y` και `x` είναι διανύσματα, τότε η `hist(y,x)` εμφανίζει ένα ιστόγραμμα των τιμών στο `y` μοιράζοντάς τις σε διαστήματα με κέντρα τα `x`. (Το μέσο του διαστήματος k είναι το `x(k)`.)

Out-of-Memory

Υπάρχει ένα όριο στο μέγεθος των πινάκων που μπορεί να διαχειριστεί το MATLAB, που εξαρτάται από την έκδοση του MATLAB και τις δυνατότητες του υπολογιστή σας. Ένα καλό, ασφαλές όριο είναι να μη δημιουργούμε πίνακες με περισσότερα από ένα εκατομμύριο στοιχείο. Αντικαθιστώντας το "10000" με "1000000000" στο `Eg6_1` θα είχε σαν αποτέλεσμα την εμφάνιση του μηνύματος "memory exceeded".

mod

Αν τα `x` και `y` έχουν τιμές με το ίδιο πρόσημο, τότε η `mod(x,y)` επιστρέφει το υπόλοιπο της `x/y`, ακριβώς όπως και η `rem(x,y)`. Αν τα `x` και `y` έχουν αντίθετα πρόσημα, τότε οι `mod` και `rem` δίνουν διαφορετικά αποτελέσματα, π.χ., το `mod(-23,7)` είναι 5 ενώ το `rem(-23,7)` είναι -2.

Λογικές Μεταβλητές

Μια λογική μεταβλητή είναι εκείνη στην οποία αποθηκεύουμε την τιμή μιας λογικής έκφρασης. Στο MATLAB, αυτό είναι είτε 0 (ψευδής) ή 1 (αληθής). Οι λογικές μεταβλητές ονομάζονται και boolean μεταβλητές.

Διανυσματικές Λογικές Πράξεις

Είναι δυνατό να εκτελέσουμε λογικές πράξεις ανάμεσα σε διανύσματα εφόσον τα διανύσματα έχουν το ίδιο μήκος και τον ίδιο προσανατολισμό. Το αποτέλεσμα είναι ένα διάνυσμα λογικού τύπου, που καλείται και boolean διάνυσμα, που αποτελείται από μηδενικά

και μονάδες. Η συγκεκριμένη τιμή καθενός από τα στοιχεία του είναι είτε 0 (ψευδής) ή 1 (αληθής) σύμφωνα με το αποτέλεσμα της πράξης σε αυτό το στοιχείο. Για διανυσματικές λογικές πράξεις πρέπει να χρησιμοποιούνται οι μορφές των τελεστών "και" και "ή" (`&,|`), οι οποίοι δεν υπολογίζονται με shortcircuiting. Για παράδειγμα,

```
a = [2 0 1]; b = [1 4 3]; c = [2 2 0];
R = (a > b) | (a > c);
S = (a < b) & (b < c);
```

any και all

Όταν εφαρμόζεται σε ένα διάνυσμα, η `any` επιστρέφει 1 αν οποιοδήποτε στοιχείο του διανύσματος είναι μη μηδενικό και επιστρέφει μηδέν στην αντίθετη περίπτωση. Έτσι αν τα `x` και `y` είναι διανύσματα, τότε η τιμή της `any(x<y)` είναι 1 αν και μόνο αν `x(k)<y(k)` για κάποιο δείκτη `k`.

Όταν εφαρμόζεται σε ένα διάνυσμα, η `all` επιστρέφει 1 αν όλα τα στοιχεία του διανύσματος είναι μη μηδενικό και επιστρέφει μηδέν στην αντίθετη περίπτωση. Έτσι αν τα `x` και `y` είναι διανύσματα, τότε η τιμή της `any(x<y)` είναι 1 αν και μόνο αν `x(k)<y(k)` για κάθε δείκτη `k`.

Ασκήσεις

Τ6.1.1 Αφαιρέστε τις εντολές `rand('seed',0)` και `randn('seed',0)` από το `Eg6_1` και παρατηρήστε πώς οι εκτιμήσεις του π αλλάζουν από τρέξιμο σε τρέξιμο του script.

Τ6.1.2 Στο `Eg6_1`, πώς επηρεάζεται η ακρίβεια του `piEstU` από την αύξηση του n;

Τ6.1.3 Στο `Eg6_1`, πώς επηρεάζεται η ακρίβεια του `piEstN` από την αύξηση της τυπικής απόκλισης `sigma`;

Τ6.1.4 Γράψτε συναρτήσεις `MyMean(x)` και `MyStd(x)` που υπολογίζουν τη μέση τιμή μ και την τυπική απόκλιση μιας κατανομής από τους τύπους

$$\mu = \frac{1}{n}\sum_{i=1}^{n} x_i, \qquad \sigma = \sqrt{\frac{1}{n-1}\sum_{i=1}^{n}(x_i-\mu)^2}.$$

Τ6.1.5 Γράψτε ένα ενιαίο πρόγραμμα που να σχεδιάζει γραφικές παραστάσεις σαν και αυτές στις Εικόνες 6.2 και 6.4.

Τ6.1.6 Γράψτε ένα ενιαίο πρόγραμμα που να σχεδιάζει γραφικές παραστάσεις σαν και αυτές στις Εικόνες 6.3 και 6.5.

Τ6.1.7 Γράψτε ένα πρόγραμμα που να δημιουργεί 10^6 τιμές από την κανονική κατανομή με μέση τιμή $\mu = 75$ και τυπική απόκλιση $\sigma = 12$. Δείξτε την κατανομή των δεδομένων σε ένα ιστόγραμμα χρησιμοποιώντας 17 διαστήματα με τα μέσα τους στο διάστημα $[-4\sigma, 4\sigma]$. (Αυτό είναι παρόμοιο με την Εικόνα 6.4.)

Τ6.1.8 Υπολογίστε τη μέση τιμή της ομοιόμορφης κατανομής υπολογίζοντας το μέσο όρο ενός δισεκατομμυρίου τιμών από τη `rand`. (Υπόδειξη: Συγκεντρώστε τα δεδομένα από 1000 κλήσεις της μορφής `rand(1000000,1)`).

Π6.1.9 Γράψτε ένα πρόγραμμα που να παράγει μια λίστα 100 ακεραίων επιλεγμένων τυχαία από το σύνολο

$$\{-20,\ -10,\ 0,\ 10,\ 20,\ 30\}.$$

Π6.1.10 Γράψτε ένα πρόγραμμα που να παράγει μια λίστα 100 πραγματικών αριθμών επιλεγμένων τυχαία από το σύνολο $\{x\ :\ 0 < x < 2$ ή $7 < x < 10\ \}$.

Π6.1.11 Ποια είναι η πιθανότητα ρίχνοντας τρεις ζαριές δύο τουλάχιστον από αυτές να φέρουν το ίδιο αποτέλεσμα; Ποια είναι η πιθανότητα το αποτέλεσμα της τρίτης ζαριάς να είναι ανάμεσα σε αυτά των δύο πρώτων; Χρησιμοποιήστε προσομοίωση για να εκτιμήσετε τις πιθανότητες.

Π6.1.12 Ας υποθέσουμε ότι οι συντελεστές του τριωνύμου $ax^2 + bx + c$ επιλέγονται από την ομοιόμορφη κατανομή στο $(-2, 2)$. Ποια είναι τότε η πιθανότητα μιγαδικών ριζών; Τι θα συμβεί αν οι συντελεστές επιλεγούν με τη `randn` με μέση τιμή $\mu = 0$ και τυπική απόκλιση $\sigma = .4$?

Π6.1.13 Ένα βελάκι εκτοξεύεται με τυχαίο τρόπο και καρφώνεται σε τετραγωνικό στόχο. Αν υποθέσουμε ότι οποιαδήποτε δύο μέρη του στόχου με ίσα εμβαδά είναι εξίσου πιθανό να χτυπηθούν, βρείτε την πιθανότητα το σημείο που θα χτυπηθεί να είναι πλησιέστερα στο κέντρο από ότι σε οποιαδήποτε άκρη του. Σημειώστε ότι η απάντηση δεν εξαρτάται από το μέγεθος του τετραγώνου.

Π6.1.14 Ένα κέρμα διαμέτρου ενός εκατοστού ρίχνεται σε επιφάνεια που είναι ένας πίνακας n-επί-n από τετράγωνα τα οποία έχουν πλευρά δύο εκατοστών το καθένα. Ποια είναι η πιθανότητα το κέρμα να βρίσκεται εξ' ολοκλήρου μέσα σε ένα τετράγωνο; Υποθέστε ότι το κέντρο του κέρματος προσγειώνεται με τυχαίο τρόπο πάνω στην επιφάνεια.

Π6.1.15 Επιλέγουμε με τυχαίο τρόπο δύο σημεία πάνω στο μοναδιαίο κύκλο. Ποια είναι η πιθανότητα η χορδή που τα ενώνει να έχει μήκος μεγαλύτερο από 1;

Π6.1.16 Ένα σημείο (x, y) επιλέγεται με τυχαίο τρόπο πάνω στο ημικύκλιο $S = \{(x, y)\ :\ x^2 + y^2 = 1, y \geq 0\}$. Ποια είναι η αναμενόμενη τιμή του εμβαδού του ορθογωνίου τριγώνου που σχηματίζεται από τα σημεία (x, y), $(1, 0)$, και $(-1, 0)$; Σημειώστε ότι η τυχαία επιλογή του (x, y) ισοδυναμεί με την επιλογή τυχαίας γωνίας θ στο $(0, \pi)$ και τον καθορισμό των συντεταγμένων του σημείου από τις σχέσεις $x = \cos(\theta)$ και $y = \sin(\theta)$.

Π6.1.17 Ένα ραβδί μοναδιαίου μήκους σπάει σε δύο κομμάτια. Ας υποθέσουμε ότι το σημείο διαχωρισμού επιλέγεται τυχαία. Κατά μέσο όρο, πόσο μακρύ είναι το κοντύτερο κομμάτι;

Π6.1.18 Έστω N ένας θετικός ακέραιος. Στο παιχνίδι "Gap N," ένα συμμετρικό κέρμα ρίχνεται διαδοχικά μέχρις ότου η διαφορά ανάμεσα στα αποτελέσματα "κορώνα" και "γράμματα" είναι N. Το σκορ είναι ο αριθμός των απαιτούμενων ρίψεων. Έτσι αν $N = 3$ και

ΚΓΓΚΓΚΓΓΚΓΓ

η αλληλουχία των αποτελεσμάτων από τις ρίψεις του νομίσματος, το σκορ είναι 11. Σημειώστε ότι

$$|\#\text{ρίψεων} - \#\text{κορωνών}| < 3$$

μέχρι την 11η ρίψη. Για δεδομένο N, Ποια είναι η αναμενόμενη τιμή (δηλαδή, η μέση τιμή) του σκορ; Ποια είναι η πιθανότητα να τελειώσει το παιχνίδι σε όχι περισσότερες από $4N$ ρίψεις; Δώστε προσεγγιστικές απαντήσεις σε αυτά τα ερωτήματα που να βασίζονται στην προσομοίωση ενός μεγάλου αριθμού παιχνιδιών.

Π6.1.19 (α) Γράψτε μια συνάρτηση `p = ProbG(L,R)` που να εκχωρεί στο `p` μια εκτίμηση του εμβαδού κάτω από τη συνάρτηση

$$f(x) = \frac{1}{\sqrt{2\pi}} e^{-x^2/2} \tag{6.2}$$

από το L στο R. (Θεωρήστε ότι $L < R$). Χρησιμοποιήστε Monte Carlo. Υπόδειξη: Ρίξτε βελάκια στο ορθογώνιο με κορυφές $(L,0)$, $(R,0)$, $(R,1)$, και $(L,1)$ και μετρήστε πόσα από αυτά είναι κάτω από την καμπύλη.

(β) Γράψτε μια συνάρτηση `p = ProbN(L,R)` που να υπολογίζει την πιθανότητα να είναι μια τιμή που παράγει η `randn` ανάμεσα στα L και R.

(γ) Γράψτε ένα πρόγραμμα που να δέχεται σαν είσοδο τα L και R και να εμφανίζει τις τιμές που επιστρέφουν οι `ProbG` και `ProbN` για αυτές.

Π6.1.20 Το επόμενο τμήμα κώδικα εκχωρεί στο `deck` ένα διάνυσμα-στήλη 52 θέσεων που είναι μια τυχαία μετάθεση των ακεραίων 1 έως 52:

```
[x,deck] = sort(rand(52,1))
```

(Θα μάθουμε για τη συνάρτηση `sort` του MATLAB στην §8.2. Για την ώρα, είναι μια μέθοδος "μαύρο κουτί" που παράγει μια τυχαία αναδιάταξη των πρώτων 52 ακεραίων). Ας συσχετίσουμε κάθε ακέραιο με ένα τραπουλόχαρτο:

1	♠ Άσσος	14	♡ Άσσος	27	♣ Άσσος	40	♢Άσσος
2	♠ Δυο	15	♡ Δυο	28	♣ Δυο	41	♢ Δυο
⋮	⋮	⋮	⋮	⋮	⋮	⋮	⋮
13	♠ Ρήγας	26	♡ Ρήγας	39	♣ Ρήγας	52	♢ Ρήγας

Χρησιμοποιώντας λογικές μεταβλητές και διανυσματικές συγκρίσεις μπορούμε να απαντήσουμε άμεσα σε ερωτήσεις σχετικές με μια "μοιρασιά". Για παράδειγμα, έστω ότι τα φύλλα σας αποτελούνται από τα τραπουλόχαρτα που αντιπροσωπεύονται από τα `deck(1:13)`. Να μια εντολή που εκχωρεί στο `nHearts` πόσες κούπες σας έχουν μοιραστεί:

```
nHearts = sum( 14 <= deck(1:13) & deck(1:13) <= 26 )
```

Να μια εντολή που υπολογίζει των αριθμό των Βαλέδων σε ένα χέρι 5 φύλλων:

```
nJacks = sum(mod(deck(1:5),13)==11);
```

(α) Στο πόκερ, ένα χέρι 5 φύλλων όπου όλα τα φύλλα είναι είναι του ιδίου χρώματος ονομάζεται "χρώμα". Γράψτε ένα τμήμα κώδικα που εκχωρεί στο `IsFlush` την τιμή 1 αν το

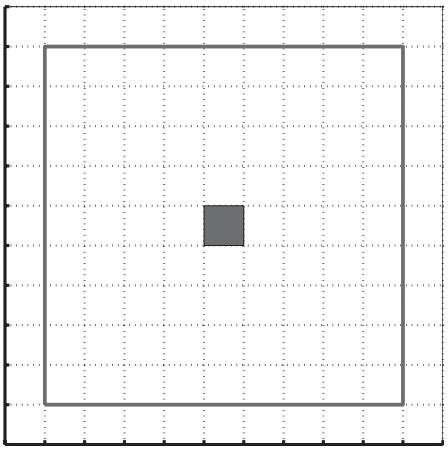
`deck(1:5)` είναι “χρώμα” και την τιμή 0 αν δεν είναι. (β) Γράψτε ένα πρόγραμμα που να υπολογίζει την πιθανότητα να είναι “χρώμα” ένα χέρι 5 φύλλων. (γ) Γράψτε ένα πρόγραμμα που να υπολογίζει την πιθανότητα να μην υπάρχουν δυο χαρτιά με την ίδια αξία σε ένα χέρι 5 φύλλων. (δ) Γράψτε ένα πρόγραμμα που να υπολογίζει την πιθανότητα να λείπει τουλάχιστον ένα χρώμα σε ένα χέρι 5 φύλλων. (ε) Γράψτε ένα πρόγραμμα που να υπολογίζει την πιθανότητα να υπάρχουν ακριβώς τρία χαρτιά της ίδιας αξίας σε ένα χέρι 5 φύλλων.

6.2 Ζάρι και Πυξίδα

Το Πρόβλημα

Έστω μια τετραγωνική περιοχή $(2n+1)$-επί-$(2n+1)$ με κέντρο στην αρχή των αξόνων που έχει καλυφθεί από τετράγωνα 1-επί-1. Δείτε την Εικόνα 6.6. Ένα ρομπότ τοποθετείται στο κεντρικό τετράγωνο και προχωράει πηδώντας από τετράγωνο σε τετράγωνο σύμφωνα με κάποιους πολύ απλούς κανόνες. Ειδικότερα, αν (x_c, y_c) είναι η τρέχουσα θέση του ρομπότ, τότε

- Με πιθανότητα $P_N = .25$ προχωράει στο “βόρειο” γείτονα του. Η νέα του θέση είναι τότε η $(x_c, y_c + 1)$.
- Με πιθανότητα $P_E = .25$ προχωράει στον “ανατολικό” γείτονα του. Η νέα του θέση είναι τότε η $(x_c + 1, y_c)$.
- Με πιθανότητα $P_S = .25$ προχωράει στο “νότιο” γείτονα του. Η νέα του θέση είναι τότε η $(x_c, y_c - 1)$.
- Με πιθανότητα $P_W = .25$ προχωράει στο “δυτικό” γείτονα του. Η νέα του θέση είναι τότε η $(x_c - 1, y_c)$.



Εικόνα 6.6. *Ένα “Πλακοστρωμένο” Τετράγωνο ($n = 5$).*

Η μεταπήδηση συνεχίζεται μέχρις ότου το ρομπότ φτάσει σε κάποιο από τα ακραία τετράγωνα. Στο παράδειγμα με $n = 5$, αυτό σημαίνει ότι η προσομοίωση τερματίζεται μόλις $|x_c| = 5$ ή $|y_c| = 5$. Μια προσομοίωση αυτού του τύπου ονομάζεται *τυχαίος περίπατος*.

Για δεδομένο n, ποιος είναι ο μέσος αριθμός μεταπηδήσεων που απαιτείται για να φτάσει το ρομπότ στο σύνορο;

Ανάπτυξη Προγράμματος

Για να διευκολυνθεί η ανάλυσή μας, θα ξεκινήσουμε υλοποιώντας μια συνάρτηση που προσομοιώνει ένα τυχαίο περίπατο στο πλέγμα και επιστρέφει τη "διαδρομή":

```
function [x y] = RandomWalk2D(n)
% n είναι ένας θετικός ακέραιος.
% Προσομοιώνει ένα τυχαίο περίπατο στο επίπεδο που
% συνεχίζεται μέχρις ότου η απόλυτη τιμή της
% συντεταγμένης x ή της συντεταγμένης y  του ρομπότ να
% είναι ίση με το n.  Τα x και y είναι διανύσματα-γραμμή
% με την ιδιότητα ότι η (x(k),y(k)) είναι η θέση του
% ρομποτ μετά από k μεταπηδήσεις, k=1:length(x).
```

Να μια ενδεικτική διαδρομή για την περίπτωση $n = 5$:

	1	2	3	4	5	6	7	8	9	10	11	12	13	14
x:	0	0	0	1	0	−1	−1	0	−1	−2	−2	−3	−3	−3
y:	−1	0	−1	−1	−1	−1	−2	−2	−2	−2	−3	−3	−4	−5

Γενικότερα, αν απαιτούνται m μεταπηδήσεις, τότε για $k = 1 : m$, τα στοιχεία `x(k)` και `y(k)` των διανυσμάτων είναι οι συντεταγμένες xy του ρομπότ *μετά* τη μεταπήδηση k. Αυτό μπορεί να γίνει με το εξής πλαίσιο:

```
% Αρχικοποίησε το μετρητή μεταπηδήσεων...
k = 0;
% Αρχικοποίησε την τρέχουσα θέση...
xc = 0; yc = 0;
while το ρομπότ δεν έχει φτάσει στο σύνορο

    Διάλεξε μια κατεύθυνση και ενημέρωσε τα xc και yc
    k = k+1
    Αποθήκευσε τα xc και yc στα x(k) και y(k).
end
```

Για να ελέγξουμε αν το ρομπότ δε βρίσκεται στο σύνορο, χρειάζεται να εξασφαλίσουμε ότι οι τιμές και της `abs(xc)` και της `abs(yc)` είναι μικρότερες του n. Έτσι, ο τυχαίος περίπατος συνεχίζεται όσο η λογική έκφραση

```
abs(xc) < n && abs(yc) < n
```

είναι αληθής.

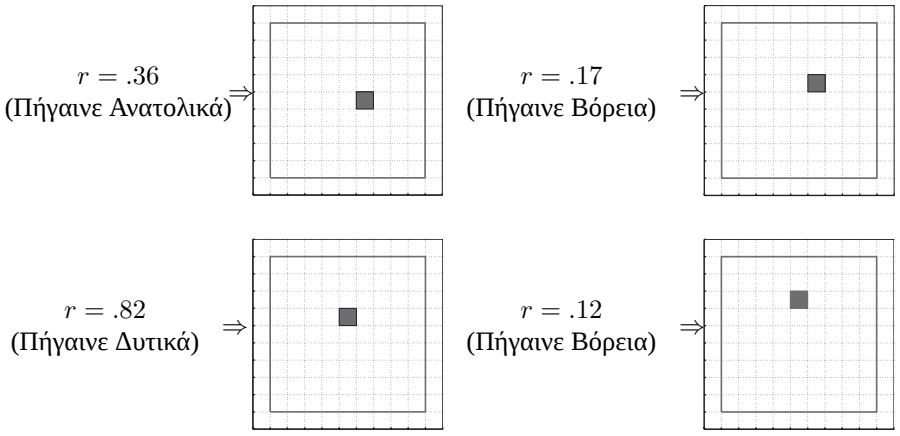
Για να προσομοιώσουμε την τυχαία επιλογή μιας μεταπήδησης προς βορά, ανατολή, νότο ή δύση, παίρνουμε μια τυχαία τιμή καλώντας τη `rand` και χρησιμοποιούμε ως εξής για να επιλέξουμε ένα από τα σημεία του ορίζοντα:

```
r = rand(1);
if r <= .25
    Πήγαινε Βόρεια
elseif .25 < r && r <= .50
    Πήγαινε Ανατολικά
elseif .50 < r && r <= .75
    Πήγαινε Νότια
else
    Πήγαινε Δυτικά
end
```

Χρησιμοποιώντας το γεγονός ότι η δομή `if-elseif` "ενεργεί" πάνω στην πρώτη από τις συνθήκες που αληθεύουν, ο παραπάνω κώδικας είναι ισοδύναμος με

```
r = rand(1);
if r <= .25
    Πήγαινε Βόρεια
elseif r <= .50
    Πήγαινε Ανατολικά
elseif r <= .75
    Πήγαινε Νότια
else
    Πήγαινε Δυτικά
end
```

Να πως θα μπορούσαν να εξελιχθούν τα πρώτα τέσσερα βήματα του τυχαίου περιπάτου:



Η τελική υλοποίηση που δίνεται παρακάτω έχει προκύψει παρατηρώντας ότι η μετακίνηση προς τα βόρεια (νότια) αντιστοιχεί σε αύξηση (μείωση) της τιμής του `yc` κατά ένα και η μετακίνηση προς τα ανατολικά (δυτικά) αντιστοιχεί με αύξηση (μείωση) της τιμής του

Η Συνάρτηση `RandomWalk2D`

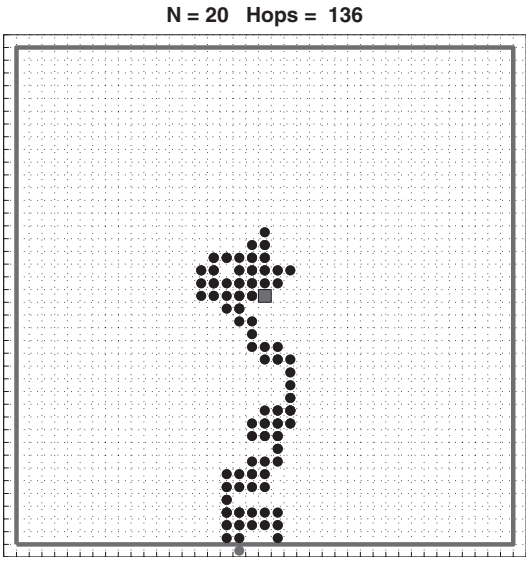
```
function [x y] = RandomWalk2D(n)
% n είναι ένας θετικός ακέραιος.
% Προσομοιώνει ένα τυχαίο περίπατο στο επίπεδο που
% συνεχίζεται μέχρις ότου η απόλυτη τιμή της
% συντεταγμένης x ή της συντεταγμένης y του ρομπότ να
% είναι ίση με το n. Τα x και y είναι διανύσματα-γραμμή
% με την ιδιότητα ότι η (x(k),y(k)) είναι η θέση του
% ρομποτ μετά από k μεταπηδήσεις, k=1:length(x).


% Αρχικοποίησε το μετρητή μεταπηδήσεων και την αρχική θέση...
k = 0; xc = 0; yc = 0;
% Ο τυχαίος περίπατος...
while abs(xc)<n && abs(yc)< n
    % Το ρομπότ στο (xc,yc), προσομοίωσε μια μεταπήδηση...
    r = rand(1);
    if r <= .25
        yc = yc+1; % Πήγαινε Βόρεια
    elseif r<=.50
        xc = xc+1; % Πήγαινε Ανατολικά
    elseif r < .75
        yc = yc-1; % Πήγαινε Νότια
    else
        xc = xc-1; % Πήγαινε Δυτικά
    end
    % Σώσε τη νέα θέση...
    k = k+1; x(k) = xc; y(k) = yc;
end
```

`xc` κατά ένα. Στην Εικόνα 6.7 εμφανίζεται μια ενδεικτική διαδρομή προς το σύνορο που παράγεται από τη `RandomWalk2D`.

Μπορούμε επίσης να χρησιμοποιήσουμε αυτή τη συνάρτηση για να λύσουμε το πρόβλημα που τέθηκε στην αρχή αυτής της ενότητας. Για να εκτιμήσουμε το μέσο "μήκος" ενός τυχαίου περιπάτου για μια δεδομένη τιμή του n, απλώς τρέχουμε τη `RandomWalk2D` πολλές φορές και υπολογίζουμε το μέσο μήκος του διανύσματος `x` (ή `y`) της διαδρομής. Δείτε σχετικά το script `Eg6_2`.

Τα αποτελέσματα υποδηλώνουν ότι τι μέσο μήκος ενός τυχαίου περιπάτου αυξάνεται όπως το τετράγωνο της τιμής του n.



Εικόνα 6.7. *Ένα Δείγμα Διαδρομής (*$n = 20$*).*

Το Script `Eg6_2`

```
% Script Eg6_2
% Υπολογίζει το μέσο αριθμό μεταπηδήσεων που χρειάζεται
% το ρομπότ για να φτάσει στο σύνορο

clc
disp('   n    Μ.Ο. προς το Σύνορο')
disp('----------------------------')
nTrials = 100;
for n = 5:5:50
    s = 0;
    for k=1:nTrials
        [x,y] = RandomWalk2D(n);
        s = s + length(x);
    end
    ave = s/nTrials;
    fprintf(' %3d         %8.3f\n',n,ave)
end
fprintf('\n\n(Τα αποτελέσματα βασίζονται σε %d δοκιμές)\n',...
         nTrials)
```

Δείγμα Εξόδου του Script `Eg6_2`

```
  n    Μ.Ο. προς το Σύνορο
---------------------------
  5           28.569
 10          120.050
 15          263.664
 20          458.544
 25          738.070
 30         1067.283
 35         1437.056
 40         1860.654
 45         2385.057
 50         3056.605

(Τα αποτελέσματα βασίζονται σε 1000 δοκιμές)
```

Συζήτηση: Random Web Surfer[10]

Οι τυχαίοι περίπατοι έχουν χρησιμοποιηθεί για τη μοντελοποίηση διαφορετικών φαινομένων σε πολλά πεδία μελέτης:

Πεδίο	Θέμα
Οικονομικά	τιμές μετοχών
Πληροφορική	ανάλυση δικτύων
Φυσική	κίνηση Brown
Γενετική Πληθυσμών	γενετική παρέκκλιση
Νευροεπιστήμη	ακολουθία πυροδότησης νευρώνων
Ψυχολογία	λήψη αποφάσεων
Αντίληψη	οφθαλμικές κινήσεις προσήλωσης

Μια από τις πιο ενδιαφέρουσες και σημαντικές εφαρμογές βρίσκεται στην καρδιά του PageRank, της μεθόδου που χρησιμοποιεί το Google για να καθορίσει τη σχετική σπουδαιότητα μιας ιστοσελίδας που στηρίζεται στη βασική δομή συνδέσμων του Διαδικτύου. Στο παρασκήνιο "κρύβεται" ένας τυχαίος περίπατος. Αντί για ένα ρομπότ που μετακινείται τυχαία από τετράγωνο σε τετράγωνο έχουμε ένα "random web surfer" που πηδάει από ιστοσελίδα σε ιστοσελίδα. Όταν ο surfer είναι σε μια ιστοσελίδα με k εξωτερικούς συνδέσμους, επιλέγει έναν από αυτούς στην τύχη και "πηγαίνει εκεί". Έτσι, αν υπάρχουν πέντε εξωτερικοί σύνδεσμοι και ένας από αυτούς αναφέρεται στην προσωπική σας ιστοσελίδα, τότε με πιθανότητα 0.2 η επόμενη στάση του surfer θα είναι εκεί. Αν δεν υπάρχουν εξωτερικοί σύνδεσμοι, τότε η επόμενη στάση του surfer καθορίζεται επιλέγοντας τυχαία μια από τις

[10] Σ.τ.Μ. 'Ενας "random web surfer" είναι ένας χρήστης του Διαδικτύου που μετακινείται από ιστοσελίδα σε ιστοσελίδα ακολουθώντας συνδέσμους που επιλέγει με τυχαία κριτήρια.

10 δισεκατομμύρια περίπου ιστοσελίδες που αποτελούν σήμερα το Διαδίκτυο. Ο αλγόριθμος PageRank προσομοιώνει αυτή τη διαδικασία για ένα μεγάλο αριθμό από random web surfers και καταγράφοντας τις μετακινήσεις τους είναι σε θέση να κατατάξει όλες τις ιστοσελίδες ανάλογα με τη σημασία τους. Η κατάταξη αυτή καθορίζει εν μέρει την απόκριση της μηχανής αναζήτησης του Google όταν υποβάλουμε ένα ερώτημα.

Επισκόπηση MATLAB

Συναρτήσεις με Περισσότερες από Μία Παραμέτρους Εξόδου

Αν μια συνάρτηση έχει περισσότερες από μία παραμέτρους εξόδου, τότε αυτές πρέπει να περικλείονται ανάμεσα σε αγκύλες:

```
[ Παράμετροι Εξόδου ] = function ( Παράμετροι Εισόδου )
```

Σε κάθε παράμετρο εξόδου στη λίστα πρέπει να εκχωρηθεί μια τιμή κατά τη διάρκεια εκτέλεσης της συνάρτησης.

Ασκήσεις

Τ6.2.1 Δεδομένου ότι τα `x` και `y` προσδιορίζουν τη διαδρομή του ρομπότ, γράψτε ένα πρόγραμμα που να εμφανίζει πόσες φορές το ρομπότ κινήθηκε σε καθεμία από τις τέσσερις κατευθύνσεις: βόρεια, ανατολικά, νότια και δυτικά.

Τ6.2.2 Τροποποιήστε τη συνάρτηση `RandomWalk2D` έτσι ώστε να μη χρησιμοποιεί την εντολή `if`. Παρατηρήστε ότι σε κάθε επανάληψη η ενημέρωση της θέσης έχει τη μορφή

$$\begin{aligned} x_{\text{new}} &= x_{\text{old}} + \triangle x \\ y_{\text{new}} &= y_{\text{old}} + \triangle y. \end{aligned}$$

Θα χρειαστείτε δυο διανύσματα, `deltaX` και `deltaY`, καθένα μήκους τέσσερα. Αρχικοποιήστε τα διανύσματα έτσι ώστε κάθε ζεύγος τιμών `deltaX(k)` και `deltaY(k)`, όπου `k` = 1, 2, 3, ή 4, αντιπροσωπεύει μια μετακίνηση σε μια από τις κατευθύνσεις. represents a move in one direction.

Τ6.2.3 Τροποποιήστε τη συνάρτηση `RandomWalk2D` για να προσομοιώνει ένα τυχαίο περίπατο που επιτρέπει μετακίνηση σε τέσσερις επιπλέον κατευθύνσεις: ΒΑ, ΝΑ, ΝΔ και ΒΔ. Και οι οκτώ κατευθύνσεις είναι εξίσου πιθανό να επιλεχθούν.

Π6.2.4 Πόσες φορές, κατά μέσο όρο, επισκέπτεται το ρομπότ το αρχικό τετράγωνο; Να σχεδιάσετε ένα ραβδόγραμμα που να εμφανίζει αυτές τις αναμενόμενες τιμές για $N = 5$, 10, 15, 20 και 25.

Π6.2.5 Εικάζεται ότι το ρομπότ είναι πιο πιθανό να εξέλθει κοντά στη μέση μιας πλευράς παρά κοντά σε κάποια γωνία. Να σχεδιάσετε ένα ραβδόγραμμα που να μας διαφωτίζει σχετικά με αυτή την εικασία.

Π6.2.6 Ορίζουμε την απόσταση d ανάμεσα στην τρέχουσα θέση (x_c, y_c) και στο σύνορο από

$$d = \min\{\ |x_c - n|,\ |x_c + n|,\ |y_c - n|,\ |y_c + n|\ \}.$$

Για δεδομένο n, να σχεδιάσετε ένα ραβδόγραμμα που να απεικονίζει την πιθανότητα, η απόσταση του ρομπότ από το σύνορο κατά τη διάρκεια της διαδρομής να είναι $1, 2, \ldots, n-1$.

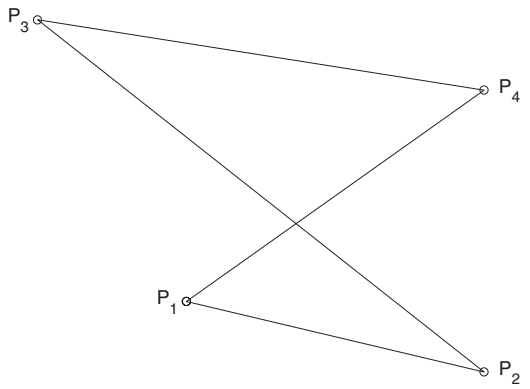
Π6.2.7 Η `RandomWalk2D` προϋποθέτει ότι καθεμία από τις πιθανότητες κατεύθυνσης p_N, p_E, p_S και p_W είναι .25. Γενικεύστε τη `RandomWalk2D` έτσι ώστε να δέχεται σαν είσοδο αυτούς τους τέσσερις αριθμούς επιπλέον του n. Είναι σαφές, ότι αν μια από τις πιθανότητες κατεύθυνσης είναι μεγαλύτερη από τις άλλες, τότε είναι πιο πιθανό το ρομπότ να εξέλθει από την αντίστοιχη πλευρά. Εξερευνήστε το συσχετισμό ανάμεσα στις πιθανότητες κατεύθυνσης και πλευρών εξόδου.

Π6.2.8 Γενικεύστε το δισδιάστατο τυχαίο περίπατο που συζητήσαμε σε αυτή την ενότητα σε τρεις διαστάσεις. Σε αυτό το καθεστώς, το ρομπότ μπορεί να μετακινηθεί βόρεια, ανατολικά, νότια, δυτικά, πάνω ή κάτω. Αντί να μετακινείται πάνω σε ένα επίπεδο πλέγμα, το ρομπότ μετακινείται πάνω σε ένα κυβικό πλέγμα. Κατά μέσο όρο και σαν συνάρτηση του n, πόσες μεταπηδήσεις χρειάζονται μέχρι να φτάσει το ρομπότ σε κάποια από τις πλευρές του κύβου;

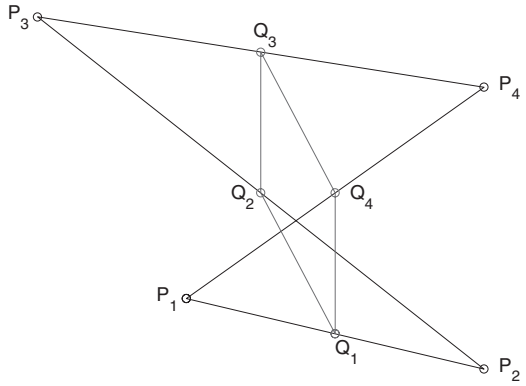
6.3 Τάξη από Χάος

Το Πρόβλημα

Αν τοποθετήσουμε τέσσερα σημεία $(x_1, y_1),\ (x_2, y_2),\ (x_3, y_3)$ και (x_4, y_4) στο επίπεδο-*xy*και τα συνδέσουμε με τη σειρά σχηματίζεται ένα τετράπλευρο. Δείτε την Εικόνα 6.8. Οργανώσαμε σκόπιμα την Εικόνα 6.8 έτσι ώστε οι πλευρές του τετραπλεύρου να "τέμνονται". Είναι ενδιαφέρον να παρατηρήσουμε ότι, αν συνδέσουμε τα μέσα των συγκεκριμένων πλευρών, τότε το "νέο" τετράπλευρο είναι *απλό*, που σημαίνει ότι οι πλευρές του δεν τέμνονται. Δείτε την Εικόνα 6.9.



Εικόνα 6.8. *Ένα Τετράπλευρο.*



Εικόνα 6.9. *Ένα Απλό Τετράπλευρο.*

Θεωρήστε ότι εφαρμόζουμε επανειλημμένα αυτή τη στρατηγική σε ένα τυχαίο πολύγωνο. Θα "ξεμπλέξουν" τελικά οι πλευρές; Πόσο θα διαρκέσει αυτή η διαδικασία σαν συνάρτηση του n, του αριθμού των πλευρών του;

Ανάπτυξη Προγράμματος

Η βασική πράξη που θα ενσωματώσουμε στην υλοποίησή μας είναι ο υπολογισμός των μέσων των πλευρών ενός δεδομένου πολυγώνου. Συνεπώς θα υλοποιήσουμε μια συνάρτηση

```
[xNew,yNew] = Smooth(x,y)
```

όπου οι τα διανύσματα `x` και `y` αποθηκεύουν τις x και y συντεταγμένες του δεδομένου πολυγώνου και τα διανύσματα `xNew` και `yNew` περιέχουν τις κορυφές του "εξομαλυμένου" πολυγώνου που προκύπτει από αυτή τη διαδικασία. Θυμηθείτε ότι αν (x_{mid}, y_{mid}) είναι το μέσο του ευθύγραμμου τμήματος που συνδέει τα (a, b) και (c, d), τότε

$$x_{mid} = (a + c)/2$$
$$y_{mid} = (b + d)/2.$$

Για να υπολογίσουμε τα μέσα των πλευρών ενός πολυγώνου με κορυφές $(x_1, y_1), \ldots, (x_n, y_n)$, πρέπει να εξάγουμε το μέσο όρο των παρακείμενων συντεταγμένων x και y. Ας εξετάσουμε την περίπτωση για $n = 5$. Οι εντολές

```
xNew(1) = (x(1) + x(2))/2; yNew(1) = (y(1) + y(2))/2;
xNew(2) = (x(2) + x(3))/2; yNew(2) = (y(2) + y(3))/2;
xNew(3) = (x(3) + x(4))/2; yNew(3) = (y(3) + y(4))/2;
xNew(4) = (x(4) + x(5))/2; yNew(4) = (y(4) + y(5))/2;
xNew(5) = (x(5) + x(1))/2; yNew(5) = (y(5) + y(1))/2;
```

εκχωρούν τις συντεταγμένες x και y των μέσων των πλευρών στα διανύσματα `xNew` και `yNew`. Για γενικότερα n πρέπει να χρησιμοποιήσουμε ένα βρόχο. Η συνάρτηση `Smooth`, που δίνεται παρακάτω, ενσωματώνει αυτές τις ιδέες.

Η Συνάρτηση `Smooth`

```
function [xNew,yNew] = Smooth(x,y)
% x και y είναι διανύσματα-στήλη με n στοιχεία που ορίζουν
% τις κορυφές ενός πολυγώνου P. Τα xNew και yNew είναι
% διανύσματα-στήλη με n στοιχεία που ορίζουν ένα πολύγωνο Q
% με κορυφές τα μέσα των πλευρών του P.

n = length(x); xNew = zeros(n,1); yNew = zeros(n,1);
for i=1:n-1
    % η νέα κορυφή i είναι μέσο πλευράς...
    xNew(i) = (x(i) + x(i+1))/2;
    yNew(i) = (y(i) + y(i+1))/2;
end
xNew(n) = (x(n)+x(1))/2;
yNew(n) = (y(n)+y(1))/2;
```

Προφανώς, η εξομάλυνση του πολυγώνου μπορεί να επαναληφθεί:

```
for k=1:nRepeat
   [x,y] = Smooth(x,y);
end
```

Για να διερευνήσουμε πόσες επαναλήψεις χρειάζονται για να καταλήξουμε σε ένα απλό πολύγωνο γράφουμε το `Eg6_3` που δημιουργεί ένα τυχαίο πολύγωνο και εμφανίζει την αλληλουχία των εξομαλύνσεων. Το script έχει διάφορα στοιχεία που είναι σημαντικό να αναδείξουμε. Πρώτον, για να διευκολύνουμε τη σύγκριση του αρχικού πολυγώνου με τις διαδοχικές εξομαλύνσεις του χρησιμοποιούμε τη `subplot` για να εμφανίζουμε και τα δύο πολύγωνα στο ίδιο παράθυρο γραφικών. Δεύτερον, μιας και ενδιαφερόμαστε μόνο για το *σχήμα*, δεν εμφανίζουμε τους άξονες και χρησιμοποιούμε για τους άξονες την επιλογή `tight` προκειμένου τα πολύγωνα που απεικονίζονται στο παράθυρο γραφικών να είναι περίπου του ιδίου μεγέθους. Τρίτον, για να προσθέσουμε κίνηση στην αλληλουχία των εξομαλύνσεων χρησιμοποιούμε την εντολή `pause(.1)` για να εμφανίζουμε κάθε εξομαλυμένο πολύγωνο για ένα κλάσμα του δευτερολέπτου. Δείτε την Εικόνα 6.10 για μερικά “στιγμιότυπα” από τη διαδικασία εξομάλυνσης ενός 20-γώνου με το script `Eg6_3`.

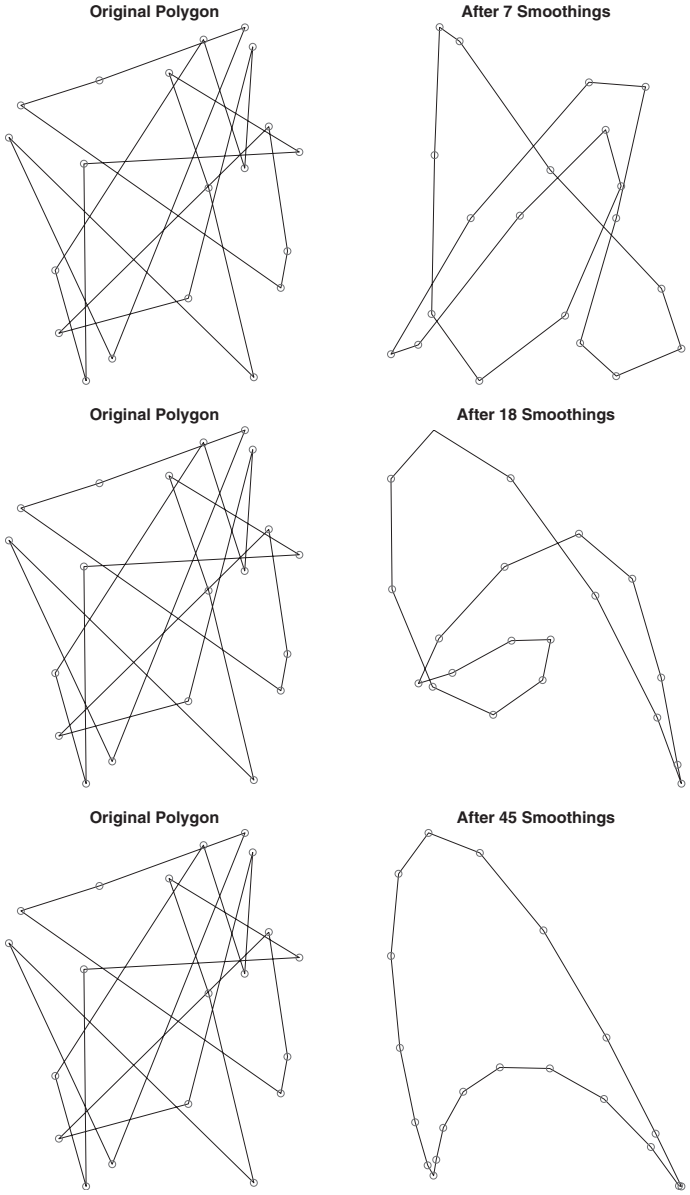
Το Script Eg6_3

```
% Script Eg6_3
% Εξομάλυνση Πολυγώνου

close all
n = input('Δώσε τον αριθμό των πλευρών του πολυγώνου: ');
nSmoothings = input('Δώσε το πλήθος των εξομαλύνσεων: ');

% Δημιούργησε και εμφάνισε ένα τυχαίο πολύγωνο
x = rand(n,1);
y = rand(n,1);
figure
set(gcf,'position',[100 100 1000 500])
subplot(1,2,1)
plot([x;x(1)],[y;y(1)],'k',x,y,'or')
axis tight off
title('Original Polygon','FontWeight','bold','FontSize',14)

% Επανειλημμένα εξομάλυνε και εμφάνισε το πολύγωνο...
for k=1:nSmoothings
    subplot(1,2,2)
    [x,y] = Smooth(x,y);
    plot([x ; x(1)],[y ; y(1)],'k',x,y,'or')
    axis tight off
    title(sprintf('After %d Smoothings',k), ...
          'FontWeight','bold','FontSize',14)
    pause(.1)
end
```



Εικόνα 6.10. *Εξομάλυνση* 20*-γώνου.*

Συζήτηση: Η Επαναλαμβανόμενη Στάθμιση Είναι Θεμελιώδης

Η επαναλαμβανόμενη στάθμιση είναι μια από τις πολύ βασικές ιδέες στους υπολογισμούς. Στην §5.1 αναπτύξαμε μια διαδικασία στάθμισης ορθογωνίων για τον υπολογισμό τετραγωνικών ριζών. Κάθε ορθογώνιο μπορεί να μετασχηματιστεί σε τετράγωνο επαναλαμβά-

νοντας αυτή την ιδέα. Σε εργαστηριακές συνθήκες, η στάθμισή επαναλαμβανόμενων μετρήσεων με κάποια διαδικασία μέσου όρου είναι μια γνωστή μεθοδολογία για την αντιμετώπιση δεδομένων με θόρυβο.

Η διαδικασία εξομάλυνσης πολυγώνων που περιγράψαμε σε αυτή την ενότητα μπορεί επίσης να θεωρηθεί ως μια διαδικασία στάθμισης. Για να φανεί αυτό, θεωρήστε την περίπτωση για $n = 5$. Το εξομαλυμένο πεντάγωνο προκύπτει από τη στάθμιση σε επίπεδο κορυφών του δεδομένου πενταγώνου

$$P\colon\ (x_1, y_1), (x_2, y_2),\ (x_3, y_3),\ (x_4, y_4),\ (x_5, y_5)$$

και εκείνου που προκύπτει με *κυκλική μετάθεση* της σειράς των κορυφών του.

$$\tilde{P}\colon\ (x_2, y_2),\ (x_3, y_3),\ (x_4, y_4),\ (x_5, y_5),\ (x_1, y_1).$$

Κάτω από την απλότητα της επαναλαμβανόμενης διαδικασίας εξομάλυνσης υπάρχουν κάποια βαθιά και πολύ δύσκολα ερωτήματα. Για παράδειγμα, καταλήγει πάντα η επαναληπτική αυτή διαδικασία σε ένα απλό πολύγωνο; Αν ναι, υπάρχει σχέση ανάμεσα στον αριθμό των απαιτούμενων εξομαλύνσεων και στον αριθμό των πλευρών του πολυγώνου; Στην §10.3 θα αναπτύξουμε "εργαλεία" που μας επιτρέπουν να αυτοματοποιήσουμε τον τερματισμό της διαδικασίας εξομάλυνσης μόλις προκύψει ένα απλό πολύγωνο. Για την ώρα, ας αρκεστούμε απλώς να "παίξουμε" με την αλληλουχία εξομάλυνσης μέσω του `Eg6_3`.

Επισκόπηση MATLAB

subplot

Με την εντολή αυτή το παράθυρο γραφικών χωρίζεται σε μικρότερα τέτοια παράθυρα:

```
subplot( # γραμμών , # στηλών , δείκτης )
```

Ο "πίνακας" των γραφικών παραθύρων έχει ένα καθορισμένο αριθμό γραμμών και στηλών. Τα παράθυρα γραφικών είναι αριθμημένα από αριστερά προς τα δεξιά και από πάνω προς τα κάτω. Το παράδειγμα

```
x = linspace(0,2*pi);
subplot(2,3,2)
plot(x,sin(x))
subplot(2,3,6)
plot(y,cos(x))
```

δημιουργεί ένα σχήμα με έξι παράθυρα γραφικών (2 γραμμές, 3 στήλες) και σχεδιάζει τη συνάρτηση ημίτονο στο παράθυρο 2 (γραμμή 1, στήλη 2) και τη συνάρτηση συνημιτόνου στο παράθυρο 6 (γραμμή 2, στήλη 3).

axis tight

Προσαρμόζει το εύρος των αξόνων x και y έτσι ώστε να είναι αρκετά μεγάλο αλλά όχι μεγαλύτερο από ότι χρειάζεται για να εμφανίσουμε τη γραφική παράσταση.

Βελτιωμένα Γραφικά

Δείτε το Παράρτημα Α για τεχνικές που μπορούν να χρησιμοποιηθούν για τη βελτίωση της ποιότητας εμφάνισης μιας γραφικής παράστασης. (Μερικές από αυτές τις τεχνικές χρησιμοποιούνται στο `Eg6_3`).

pause(s)

Η εντολή αυτή αναστέλλει την εκτέλεση για s δευτερόλεπτα. Μπορούμε με ασφάλεια να χρησιμοποιήσουμε χρόνους της τάξης του .01 δευτερολέπτου.

Ασκήσεις

Τ6.3.1 Τροποποιήστε την υλοποίηση της `Smooth` έτσι ώστε ο βρόχος να τρέχει από το ένα μέχρι και το n, αποφεύγοντας την ανάγκη να υπολογίσουμε ξεχωριστά το n-οστό ενδιάμεσο σημείο.

Τ6.3.2 Το κέντρο βάρους ενός πολυγώνου με κορυφές $(x_1, y_1), \ldots, (x_n, y_n)$ είναι το σημείο $(\bar{x}, \bar{y})$, όπου $\bar{x}$ είναι ο μέσος όρος των x_i και $\bar{y}$ είναι ο μέσος όρος των y_i. Γράψτε μια συνάρτηση `ShowPolygon(x,y)` που να σχεδιάζει το πολύγωνο που ορίζεται από τα διανύσματα `x` και `y` και να εμφανίζει στο ίδιο παράθυρο τα ευθύγραμμα τμήματα που συνδέουν κάθε κορυφή του με το κέντρο βάρους.

Τ6.3.3 Γράψτε μια συνάρτηση `[x,y] = GenPoly(n)` που να επιστρέφει στα `x` και `y` τις κορυφές του πολυγώνου που έχει n κορυφές τοποθετημένες τυχαία πάνω στο μοναδιαίο κύκλο. Για δεδομένο n, ποια είναι η μέση περίμετρος ενός τέτοιου πολυγώνου;

Π6.3.4 Έστω ότι "περπατάμε" από το σημείο (a, b) στο σημείο (c, d) κατά μήκος του ευθύγραμμου τμήματος που τα ενώνει. Αν το λ ικανοποιεί τη σχέση $0 \leq \lambda \leq 1$, τότε αναφερόμαστε στο $(a+\lambda(c-a), b+\lambda(d-b))$ σαν το σημείο λ. Σημειώστε ότι αν $\lambda = 0$, τότε βρισκόμαστε στο (a, b) ενώ αν $\lambda = 1$, τότε βρισκόμαστε στο (c, d). Το μέσο αντιστοιχεί σε $\lambda = 1/2$. Για δεδομένο λ το οποίο ικανοποιεί τη σχέση $0 < \lambda < 1$, πως επηρεάζεται η επαναληπτική διαδικασία εξομάλυνσης αν διαλέξουμε το σημείο λ κάθε πλευράς αντί για το μέσο της.

Π6.3.5 Γράψτε μια συνάρτηση `r = DistToReg(x,y)` που να επιστρέφει το λόγο των μηκών της μικρότερης προς τη μεγαλύτερη πλευρά ενός πολυγώνου που ορίζεται από τα διανύσματα `x` και `y`. Πως μεταβάλλεται αυτός ο λόγος κατά τη διάρκεια της διαδικασίας εξομάλυνσης;