fl1.m

1. Ellenőrzi, hogy a bemenő vektor (input\_vector) nem lehet üres (a isempty függvény segítségével).
2. Ellenőrzi, hogy a mantissza koordinátái a {0,1} halmazból származnak-e (az any függvénnyel és logikai kifejezésekkel).
3. Ellenőrzi, hogy az utolsó elem a vektorban az előjelbit reprezentálja, és hogy az értéke 0 vagy 1.
4. Kiszámolja a mantissza hosszát, ami a bemenő vektor hosszából levon egyet a karakterisztika és egyet az előjelbit miatt.
5. Kiszámolja az expoenst a mantissza hosszából, ami egy előre meghatározott érték (mantissa hossz - 1) alapján.
6. Kiszámolja a gépszámot az előjellel és az expoenstől függően. Ha az előjelbit 0, akkor pozitív számot számol ki, különben negatívat. Az eredmény decimális szám adja vissza.

A függvény tehát egy gépi számot számol ki a bemenő paraméterként megadott vektor alapján, figyelembe véve az előjelbit és az expoenst.

Futtatás:

result = fl1([1, 0, 1]);

fl2.m

1. Ellenőrzi, hogy a bemenő adatok megfelelő formátumúak-e, és a megfelelő értéktartományba esnek-e (pozitív egész t, egész k1 és k2, valamint k1 kisebb, mint k2).
2. Kiszámítja a gépi számhalmaz elemszámát a t érték alapján.
3. Kiszámítja a "0" és "1" nevezetes elemeket a fl1 függvény segítségével a megfelelő számokkal és nullákkal.
4. Kihasználja a gépi számhalmaz szimmetriáját a számolásokhoz. Kiszámítja az intervallumok szélességét, majd a számokat a valós számegyenesen megjeleníti, kiírva a megfelelő intervallumokat az elemekkel.

Ez a kód kiszámolja és ábrázolja a gépi számokat a valós számegyenesen, kihasználva a szimmetriát a hatékony számolás érdekében.

Futtatás: fl2(3, -1, 4);

fl3.m

1. Ellenőrzés: A kód ellenőrzi, hogy legalább egy bemenő paramétert megadtak-e, majd eldönti, hogy az előírt gépi számhalmazon kell-e dolgozni vagy egy felhasználó által megadott halmazon.
2. Alapértelmezett számhalmaz: Ha csak egy bemenő paramétert adnak meg, akkor az alapértelmezett gépi számhalmazzal dolgozik (M(8;-5;5)).
3. Ellenőrzés a gépi számhalmaz érvényességére: Ellenőrzi, hogy a megfelelő gépi számhalmaz érvényes-e (pozitív egész t, k1 és k2, valamint k1 kisebb, mint k2).
4. Mantissza hosszának számítása: Kiszámítja a mantissza hosszát a t értékből.
5. Karakterisztika számítása: Számolja ki a karakterisztikát a megadott k1 és k2 értékek alapján.
6. Előjeles mantissza számítása: Meghatározza az előjeles mantissza a k1 érték alapján.
7. Gépi alak elkészítése: Összeállítja a gépi alakot a vektorban, az utolsó eleme a karakterisztikát, a többi pedig az előjeles mantissza.
8. Az eredmény kiírása 10-es számrendszerben: Kiírja az eredményt a konzolra a gépi alakban, 10-es számrendszerben.

Futtatás:

fl3(6, -3, 3);

fl4.m

1. Először is ellenőrzi, hogy a két vektor (gépi szám) azonos számhalmazból való-e, vagyis ugyanolyan hosszúságúak és ugyanazt a karakterisztikát (az utolsó bitet) használják-e. Ha a két vektor különböző hosszúságú vagy különböző karakterisztikával rendelkezik, hibát dob.

2. Ellenőrzi, hogy a vektorok előjelet (pozitív vagy negatív) mutatnak-e. Ha bármelyik vektor első bitje 1 (negatív előjel), akkor a kód kivonást végez. Ellenkező esetben az összeadást végzi.

3. A gépi összeadás vagy kivonás során a kód végiglépked a vektorok minden elemén (a legkisebb helyiértéktől a legnagyobbig), és a műveleteket elvégzi, figyelembe véve az esetleges átviteleket (carry vagy borrow).

4. Ha a műveletek során túlcsordulás (carry vagy borrow) történik, akkor a kód az eredményhez hozzáadja vagy levonja a karakterisztika értékét, hogy a túlcsordulás kezelésre kerüljön.

5. Végül a kód normalizálja az eredményt, hogy az ne tartalmazzon túlcsordulásokat vagy felesleges biteket. Ha az eredményt ki kell egészíteni egy új helyiértékkel a túlcsordulás miatt, akkor a karakterisztika értékét módosítja.

A végén az "fl4" függvény visszaadja az összegzett vagy kivont gépi számot, amely tartalmazza az előjelbitet (az első bitet) és a karakterisztikát (az utolsó bitet), valamint a mantisszát a köztes helyiértékeken.

Futtatas:

vec1 = [1 1 0 1 0]; % első vektor

vec2 = [0 1 1 0 0]; % második vektor

result = fl4(vec1, vec2); % Az eredményt tárolja a "result" változóban

gaussel1.m

1. A függvény bemeneti paraméterként egyenletrendszer mátrixot (A) és jobboldal vektort (b) kap.

2. Ellenőrizzük, hogy az A mátrix négyzet alakú-e (méretei megegyeznek-e).

3. Létrehozunk egy augmentált mátrixot, amely az egyenletrendszer mátrixot (A) és a jobboldal vektort (b) tartalmazza.

4. Iterálunk a mátrix sorain (k sorindexel), kezdve az első sorsorral, majd folytatjuk a következő sorokkal.

5. Kiválasztjuk a pivotsort. A pivot sor a k sor és k oszlop felett található, és a legnagyobb abszolút értékű elemet tartalmazza.

6. Pivot sor kiválasztása után sorcserét hajtunk végre a kiválasztott pivotsor és a jelenlegi sor között.

7. Normalizáljuk a pivot oszlopot, úgy hogy a pivot elem értékével osztjuk az egész oszlopot. Így a pivot elem 1 lesz.

8. Eliminációt végzünk, azaz a következő sorokban nullára hozzuk azon elemeket, amelyek az adott oszlop alatt vannak.

9. Ismétlünk a következő sorra.

10. A visszavezetés (back-substitution) során kiszámoljuk a megoldást az egyenletrendszer végleges felső háromszöges formájából indulva, és fokozatosan visszavezetjük a megoldást.

11. A végén a megoldást (x vektor) adjuk vissza a függvény eredményeként.

Ez a Gauss-eliminációs algoritmus használható lineáris egyenletrendszerek megoldására. A folyamat során olyan lépéseket hajtunk végre, amelyekkel a mátrixot fokozatosan átalakítjuk felső háromszöges alakra, majd a visszavezetés lépéseivel kiszámoljuk a megoldást.

Futtatás:

A = [2, -1, 1; -3, 2, 2; 1, 1, 3];

b = [8; -9; 4];

x = gaussel1(A, b);

gaussel2.m

1. A függvény megkapja a következő bemeneti paramétereket: `A` (egyenletrendszer mátrix), `b` (jobboldal vektor), és opcionális `full\_pivoting` (teljes föelemkiválasztás) paraméter.

2. Ellenőrzi, hogy az `A` mátrix négyzet alakú-e. Ha nem, hibát dob.

3. Ha a `full\_pivoting` paraméter nincs megadva, akkor az alapértelmezett értéke `false`, ami a részleges föelemkiválasztást jelenti.

4. Létrehozza az augmentált mátrixot, amely tartalmazza az `A` mátrixot és a `b` vektort.

5. Iterál az egyenletrendszeren, és a `full\_pivoting` paraméter alapján választja ki a részleges vagy teljes föelemkiválasztást. Ha a kiválasztott módszerrel nem lehet pivotot találni, akkor automatikusan vált a másik módszerre.

6. Sorcserékkel és normalizációval hajt végre eliminációs lépéseket a kiválasztott pivot sor és oszlop alapján.

7. Végrehajtja az eliminációt, és közbeeső mátrixokat hoz létre, figyelembe véve a föelemkiválasztások hatását.

8. Visszavezeti (back-substitution) a megoldást, és kiszámolja a `x` megoldásvektort.

9. Visszaadja a `x` megoldást.

10. A kód közli a felhasználóval, ha a részleges föelemkiválasztás nem vezet eredményre, majd automatikusan vált a teljes föelemkiválasztásra.

11. Az elimináció lépéseit és a közbeeső mátrixokat is kiírja a MATLAB Command Window-ban, ami segít a műveletek nyomon követésében.

A `gaussel2` függvény tehát egy Gauss-eliminációt végző kód, amely rugalmasan kezeli a föelemkiválasztást, és figyelembe veszi a felhasználó döntéseit és a megoldás módosításait a folyamat során.  
  
  
Futtatás:

A = [2, -1, 1; -3, 2, 2; 1, 1, 3];

b = [8; -9; 4];

full\_pivoting = false; % Részleges föelemkiválasztás

x = gaussel2(A, b, full\_pivoting);

gaussel3.m

1. A függvény deklarálása, bemeneti és kimeneti argumentumok megadásával.

2. Az `A` mátrix dimenzióinak meghatározása a `size` függvény segítségével.

3. Ellenőrzi, hogy a bemeneti mátrix négyzet alakú-e (azonos sorok és oszlopok száma).

4. Ellenőrzi, hogy az üres mátrixnak nincs inverze vagy determinánsa.

5. Ellenőrzi, hogy a függvény pontosan egy bemeneti argumentumot kap.

6. Az LU-felbontás és determináns számítás elvégzése a `gausselim` függvény segítségével.

7. Ellenőrzi, hogy a determináns nem 0, különben hibát dob.

8. Az inverz mátrix számítása az LU-felbontás és determináns alapján.

9. Az identitás mátrix létrehozása, amely sorban veszi az inverz számításához szükséges vektorokat.

1. Az inverz számítása az LU-felbontás és az identitás mátrix segítségével. Az inverz mátrixot visszaadja a kimeneti változóként.

Futtatás:

A = [1, 2, 3; 0, 1, 4; 5, 6, 0];

[inverse\_matrix, determinant] = gaussel3(A);

Gramschmidt

A kód, amit "gramschmidt.m" néven mentettél el, QR-felbontást végez Gram-Schmidt ortogonalizációval egy adott mátrixon.

1. Először a kód meghatározza a bemeneti mátrix `A` méretét (sorok és oszlopok száma), azaz a `m` és `n` értékeket.

2. Ellenőrzi, hogy `A` legalább annyi oszlopot tartalmaz, mint amennyi sor van benne. Ha a mátrix oszlopainak száma kisebb, mint a soroké, akkor hibát dob, mivel QR-felbontást csak négyzetes vagy alacsony oszlopszámú mátrixokon lehet elvégezni.

3. Inicializálja az ortogonális mátrixot `Q` és a felső háromszög mátrixot `R` nullmátrixokként.

4. Egy cikluson keresztül iterál az oszlopokon (`j`), és minden oszlopot külön kezel:

a. Az aktuális oszlopot (az `A` mátrix `j`-edik oszlopát) másolja egy ideiglenes változóba `v`.

b. Egy belső cikluson keresztül iterál az oszlopokon (`i`) az aktuális oszloptól balra (`j-1`-ig). Ebben a ciklusban kiszámítja a `R` mátrix elemeit a Gram-Schmidt ortogonalizációval, és frissíti az `v` értékét.

c. A `R` mátrix diagonális elemeit kiszámítja a `v` normájával (a `norm` függvény segítségével). Ha az `R` diagonális eleme 0, akkor hibát dob, mivel ez azt jelentené, hogy az `A` oszlopai lineárisan függők, és a QR-felbontás nem lehetséges.

d. Az `A` mátrix `j`-edik oszlopát normalizálja, és az eredményt az `Q` mátrixba írja.

5. Miután befejezte az iterációt az összes oszlopon, a kód visszaadja az `Q` és `R` mátrixokat, amelyek az `A` mátrix QR-felbontása.

A kód a QR-felbontást hajtja végre a Gram-Schmidt ortogonalizációs eljárással, és ellenőrzi, hogy az eredmény valid-e (az oszlopok lineárisan függetlenek-e). Az `Q` és `R` mátrixok az eredményeket tartalmazzák.

Futtatas:

A = [1 2 3; 4 5 6; 7 8 9];

[Q, R] = gramschmidt(A);

Householder

A "householder.m" függvény kiszámolja a Householder-transzformáció mátrixát két vektor alapján: egy pont vektort (`P`) és annak képét (`P0`). A Householder-transzformáció egy ortogonális mátrix, amelyet gyakran használnak lineáris algebrai feladatokban, például mátrix diagonalizálásban vagy lineáris egyenletrendszerek megoldásában.

1. Ellenőrzi, hogy a bemeneti vektorok (`P` és `P0`) hossza azonos. Az algoritmus csak azonos hosszú vektorokkal működik.

2. Kiszámolja a Householder-transzformáció vektorát (`v`), amely a kép és a pont különbsége.

3. Normalizálja a Householder-transzformáció vektort (`v`) úgy, hogy az hossza 1 legyen. Ezt a `norm` függvény segítségével végzi el.

4. Kiszámolja a Householder-transzformáció mátrixát (`H`). Ez a mátrix egy identitásmátrixból (az azonosító mátrixból) kivonja a Householder-transzformáció vektort és annak transzponáltjának skalárszorzatából származó mátrixot.

5. A Householder-transzformáció mátrixához választja meg a paraméter előjelét a pont és a kép közötti skalárszorzat alapján. Ha a skalárszorzat negatív, akkor a mátrix negatív előjelű lesz.

Összességében a "householder" függvény egy Householder-transzformáció mátrixát készíti el a megadott pont és kép vektorok alapján, figyelembe véve a paraméter előjelét.

Futtatás:

P = [1; 2; 3];

P0 = [4; 5; 6];

H = householder(P, P0);

hhalg.m

A `hhalg` függvény a Householder-transzformáció segítségével végzi el a QR-felbontást egy négyzetes mátrixon.

1. `function [Q, R] = hhalg(A)` - A függvény definiálása, amely két kimeneti értéket, `Q`-t (ortogonális mátrix) és `R`-t (felső háromszög mátrix) ad vissza, és egy bemeneti négyzetes mátrixot (`A`) kap.

2. `[m, n] = size(A)` - Meghatározzuk a bemeneti mátrix sorainak (`m`) és oszlopainak (`n`) számát.

3. `if m < n` - Ellenőrizzük, hogy a bemeneti mátrixnak több oszlopa van-e, mint sorai. Ha nem, hibát dob, mivel a QR-felbontáshoz több oszlopra van szükség.

4. `Q = eye(m)` - Inicializáljuk az `Q` ortogonális mátrixot az egység mátrixszal. Kezdetben ez az identitás mátrix lesz.

5. `R = A` - Inicializáljuk az `R` felső háromszög mátrixot a bemeneti mátrix (`A`) másolatával.

6. `for k = 1:n` - Ciklus, amely az oszlopokon iterál (`n` számú oszlop van).

a. `x = R(k:m, k)` - Kiválasztjuk az aktuális oszlop részét, amelyet felhasználunk a Householder-transzformációhoz.

b. `e = zeros(length(x), 1)` - Létrehozunk egy nullmátrixot, amely az `e` vektor első eleme kivételével nullákat tartalmaz. Az `e` vektor az identitás vektor része.

c. `v = householder(x, e)` - Számoljuk ki a Householder-transzformáció vektorát a `householder` függvény segítségével.

d. `P = eye(m)` - Inicializáljuk a Householder-mátrixot az egység mátrixszal.

e. `P(k:m, k:m) = eye(m - k + 1) - 2 \* v \* v'` - Kiszámoljuk a Householder-mátrixot, amelyet alkalmazunk a bemeneti mátrix `R`-re. Ez az átalakítás lépésről lépésre alakítja a `R` mátrixot felső háromszög alakúvá.

f. `R = P \* R` - Az `R` mátrix átalakítása a Householder-mátrix segítségével.

g. `Q = Q \* P'` - Az `Q` mátrix átalakítása a transzponált Householder-mátrix segítségével.

7. A függvény visszaadja az ortogonális (`Q`) és a felső háromszög (`R`) mátrixokat, amelyek a bemeneti mátrix (`A`) QR-felbontását reprezentálják.

Ez a kód lépésről lépésre végzi el a QR-felbontást Householder-transzformációval egy négyzetes mátrixon.  
  
Futtatás:

% Példa egy négyzetes mátrixra

A = [1, 2, 3; 4, 5, 6; 7, 8, 9];

% Hívjuk meg a hhalg függvényt a QR-felbontás elvégzéséhez

[Q, R] = hhalg(A);

% Kiírjuk az eredményeket

disp('Q (ortogonális mátrix):');

disp(Q);

disp('R (felső háromszög mátrix):');

disp(R);

hhgraph.m

Ez a kód egy grafikus alkalmazást hoz létre, amely lehetővé teszi a felhasználó számára, hogy bekérjen két vektorot (pontot) és rajzolja meg azokat egy ábrán, valamint alkalmazza a Householder-transzformációt a két vektorra a hipersík létrehozásához, majd megjeleníti a tükrözött pontot.

1. Először a felhasználótól bekéri az eredeti pont és a hipersík irányvektorának koordinátáit.

2. Definiálja a `P` és `P0` vektorokat az adott koordinátákkal és meghívja a `householder` függvényt a Householder-transzformáció elvégzéséhez.

3. Létrehozza a grafikus ábrát (`figure`) és rajzolja meg a hipersíkot a `quiver` függvény segítségével. Az eredeti pontot és a tükrözött pontot a `scatter` függvényekkel ábrázolja.

4. Beállítja az ábra paramétereit, például az egyenlő arányú tengelyeket, címet és tengelyfeliratokat.

Ezután a kód megjeleníti az ábrát, amely tartalmazza az eredeti pontot, a hipersíkot és a tükrözött pontot. A felhasználó az ábrán láthatja az eredeti és a tükrözött pontok elhelyezkedését a hipersíkhoz képest.

Futtatás:

Kérem adja meg az eredeti pont x-koordinátáját: 2

Kérem adja meg az eredeti pont y-koordinátáját: 3

Kérem adja meg a hipersík irányvektorának x-koordinátáját: 1

Kérem adja meg a hipersík irányvektorának y-koordinátáját: 1

Lagrangeip

1. Ellenőrzi, hogy az `x\_points` és `y\_values` bemeneti vektorok mérete egyezik-e. Ha nem, hibát dob.

2. Kiszámolja az interpolációs polinomot Lagrange alakban a következő módon:

- Végigiterál az alappontokon (`x\_points`).

- Minden alappont esetén kiszámolja a Lagrange alapfüggvényt, amely az adott pontokon értékeket vesz fel.

- Az egyes Lagrange alapfüggvények szorzatait összeadja, súlyozva azokat a megfelelő függvényértékekkel (`y\_values`).

3. Grafikusan ábrázolja az eredeti adatpontokat (`x\_points`, `y\_values`) piros körökkel, majd az interpolációs polinomot kék vonallal ugyanazon az ábrán.

Ezáltal a függvény segítségével vizualizálható, hogyan illeszkedik az interpolációs polinom az adatpontokhoz.

Futtatás:

x\_points = [1, 2, 4];

y\_values = [3, 1, 2];

lagrangeip(x\_points, y\_values);

newtonip

Bemeneti adatok kezelése:

- Ha nincsenek bemeneti paraméterek megadva, a felhasználótól kéri az alappontokat (`x\_points`) és a függvényértékeket (`y\_values`).

- Ellenőrzi, hogy az alappontok és a függvényértékek száma egyezik-e. Ha nem, hibát dob.

Newton interpolációs polinom számítása:

- Számolja ki a Newton interpolációs polinom együtthatóit a megadott alappontok és függvényértékek alapján.

- A polinomot kiszámolja a Newton alakban.

Grafikus ábrázolás:

- Grafikusan ábrázolja az eredeti adatpontokat (`x\_points`, `y\_values`) piros körökkel.

- Ábrázolja az interpolációs polinomot a kiszámolt Newton alakban kék vonallal.

Új alappont hozzáadása:

- Megkérdezi a felhasználót, hogy szeretne-e új alappontot hozzáadni.

- Ha igen, kéri az új alappontot (`new\_x`) és a függvényértéket az új alappontban (`new\_y`).

- Hozzáadja az új alappontot a listához, majd újra futtatja a függvényt a frissített adatokkal.

Futtatás

% Teszthívás

x\_points = [1, 2, 4];

y\_values = [3, 1, 2];

% Hívjuk meg a függvényt

newtonip(x\_points, y\_values);

**lnmaprox.m**

A kód egy függvényt definiál, amely a legkisebb négyzetek módszerével approximál egy polinomot a megadott csomópontok alapján.

Függvénydefiníció:

- lnmaprox függvény, amely a legkisebb négyzetek módszerével approximál egy polinomot.

Bemeneti paraméterek:

- `degree`: Az approximációs polinom fokszáma.

- `nodes`: A csomópontok koordinátái egy mátrixban.

Hibaellenőrzések:

- Ellenőrzi, hogy a `nodes` mátrixnak két oszlopnak kell lennie, különben hibát dob.

- Ellenőrzi, hogy a `degree` értéke pozitív egész szám, különben hibát dob.

Csomópontok kinyerése:

- Kinyeri a csomópontok x- és y-koordinátáit két külön változóba.

Vandermonde mátrix létrehozása:

- Létrehoz egy Vandermonde mátrixot a csomópontok x-koordinátái alapján.

Csökkentett normál egyenletrendszer létrehozása:

- Kiszámolja a transzponált Vandermonde mátrix szorzatát önmagával (`AtA`) és a transzponált Vandermonde mátrix és a `y` vektor szorzatát (`AtY`).

Egyenletrendszer megoldása:

- Megoldja az egyenletrendszert az `AtA \ AtY` kifejezéssel, és ezzel kapja meg az approximációs polinom együtthatóit (`coefficients`).

Grafikon készítése (opcionális):

- Megkérdezi a felhasználót, hogy készítsen-e grafikont.

- Ha igen, kiszámolja az approximációs polinom értékeit és kirajzolja a csomópontokat és az approximációs polinomot egy grafikonon.

Visszatérési érték:

- `coefficients`: Az approximációs polinom együtthatói.

A kód a legkisebb négyzetek módszerével közelít egy polinomot a megadott csomópontok alapján, és visszaadja az approximációs polinom együtthatóit. Opcionálisan készít egy grafikont a csomópontokról és az approximációs polinomról.

Futtatás:

% Példa csomópontokra és approximációs polinom készítésére

degree = 2;

nodes = [1, 2; 2, 1; 3, 3; 4, 2];

coefficients = lnmaprox(degree, nodes);

disp('Approximációs polinom együtthatói:');

disp(coefficients);

geninv.m

A kód egy függvényt definiál, amely mátrix általánosított inverzét számolja ki rangfaktorizáció segítségével.

Függvénydefiníció:

- `geninv` függvény, amely mátrix általánosított inverzét számolja ki.

Bemeneti paraméterek:

- `A`: A bemeneti mátrix, amelynek az általánosított inverzét kiszámoljuk.

Rangfaktorizáció:

- Meghatározza a bemeneti mátrix rangját (`r`) a `rank` függvénnyel.

Mátrix méreteinek lekérése:

- Lekéri a bemeneti mátrix sorainak (`m`) és oszlopainak (`n`) számát.

Ranghiány ellenőrzése:

- Ellenőrzi, hogy a mátrix rangja kisebb-e a sorok vagy oszlopok számánál. Ha igen, hibát dob, mert a mátrixnak teljes rangúnak kell lennie az általánosított inverz kiszámításához.

QR faktorizáció:

- QR faktorizációt végez a mátrixon a `qr` beépített függvény segítségével.

Általánosított inverz kiszámítása:

- Kiszámolja az általánosított inverz mátrixot a QR faktorizáció eredményei alapján a `R \ Q'` kifejezéssel.

Visszatérési érték:

- `Aplus`: Az általánosított inverz.

A kód egy teljes folyamatot valósít meg a mátrix általánosított inverz kiszámítására rangfaktorizációval.  
  
  
Futtatás:

% Példa mátrix létrehozására

A = rand(3, 3);

% Mátrix általánosított inverzének kiszámítása

Aplus = geninv(A);

Numint.m

A kód egy fő függvényt és három al-függvényt tartalmaz, amelyek klasszikus kvadratúra formulákat alkalmaznak.

Fő Függvény

-Bemeneti paraméterek:

- integrand: Az integrandus függvényt leíró karakterlánc (string).

- a, b: Az integrálás intervallumának alsó és felső végpontjai.

- n: Az osztópontok száma.

- type: A kvadratúra formula típusa ('téglalap', 'trapéz', 'simpson').

- Működés:

- Átalakítja az integrandus karakterláncot egy inline függvénnyé.

- Létrehozza az osztópontokat a megadott intervallumban.

- Kiválasztja a megfelelő kvadratúra formulát (rectangleRule, trapezoidalRule, simpsonsRule) a type paraméter alapján.

- Visszaadja az integrál eredményét.

rectangleRule Al-függvény

- Bemeneti paraméterek:

- f: Az integrálandó függvény inline formában.

- x: Az osztópontok vektora.

- Működés:

- Kiszámolja a téglalap szabály segítségével az integrált.

trapezoidalRule Al-függvény

- Bemeneti paraméterek:

- f: Az integrálandó függvény inline formában.

- x: Az osztópontok vektora.

- Működés:

- Kiszámolja a trapéz szabály segítségével az integrált.

simpsonsRule Al-függvény

- Bemeneti paraméterek:

- f: Az integrálandó függvény inline formában.

- x: Az osztópontok vektora.

- Működés:

- Kiszámolja a Simpson szabály segítségével az integrált.

A kód általános módszert biztosít az integrálásra, és különböző kvadratúra formulákat alkalmazhatunk a kívánt pontosság eléréséhez.

Futtatás:  
integrand = 'sin(x)';

a = 0;

b = pi;

n = 100;

type = 'simpson';

result = numint(integrand, a, b, n, type);

disp(['Az integrál eredménye: ' num2str(result)]);

affin1.m

A kód egy affin transzformáció mátrixát számolja ki két adott pont alapján.

1. Ellenőrzi, hogy hány bemeneti paramétert kapott.

- Ha nincs bemeneti paraméter, a felhasználótól két pontot kér be egy grafikus felületen.

- Ha két bemeneti paraméter van megadva, ezeket az értékeket használja.

- Ha más számú bemeneti paraméter van megadva, hibát dob.

2. A kódban definiál egy alap mátrixot `P`-t, amely a két origo pontot tartalmazza: `[0 1; 1 0]`.

3. A bemeneti pontok alapján létrehoz egy képmátrixot `Q`-t.

4. Számolja a transzformáció mátrixát `A`-t, amelyet a `Q / P` művelettel kap meg.

5. Kiírja az eredményül kapott affin transzformáció mátrixot a konzolra.

A kód egy affin transzformáció mátrixának meghatározása két pont alapján, és ezt a mátrixot kiírni a konzolra

Futtatás:

affin1;

affin2.m

A függvény egy affin transzformáció mátrixát számolja ki három pont és az ezekhez tartozó transzformációbeli képek alapján. A függvény lehetőséget biztosít a bemeneti paraméterek grafikus megadására is.

1. Kezeli a bemeneti paramétereket:

- Ha nincs bemeneti paraméter, akkor a felhasználótól három csúcsot és a hozzájuk tartozó transzformációbeli képeket kéri be egy grafikus felületen.

- Ha hat bemeneti paraméter van megadva, akkor ezeket használja a függvény.

- Egyéb esetben hibát dob.

2. Készít egy grafikus felületet, ahol a felhasználó kiválaszthat három csúcsot és a hozzájuk tartozó transzformációbeli képeket. Az adatokat egy grafikonon megjeleníti.

3. Az eredeti és a transzformált csúcsokat tartalmazó mátrixokat (`P` és `Q`) létrehozza.

4. Kiszámolja az affin transzformáció mátrixát a `Q / P` művelettel.

5. Kiírja az affin transzformáció mátrixát a konzolra.

A függvény lehetővé teszi a felhasználó számára, hogy grafikusan válassza ki a pontokat és azok transzformációbeli képeit, vagy közvetlenül megadja ezeket a paramétereket. Az affin transzformáció mátrixa az eredményül kapott `A` mátrixban található, amit a függvény kiír a konzolra.

Futtatás:

affin2;