**אינטגרציה נומרית**

**מוטיבציה**: המטרה של אינטגרציה נומרית היא בדרך כלל להעריך את השטח הכלוא מתחת לעקום של פונקציה נתונה f(x) בצורה כזו או אחרת. השיטות הנומריות קיימות להתמודד עם שני מצבים שהחדו"א לא מספק פתרון ראוי:

* גם אם f(x) ידוע ומורכב מצירוף של מפונקציות אלמנטריות, את פונקצית האינטגרל הלא מסוים (האנטי נגזרת) לא תמיד (אולי אפילו בדרך כלל לא) ניתן להציג כמורכב מצירוף של פונקציות אלמטריות, כלומר לא קיים צירוף כזה שהנגזרת שלו היא f(x). דוגמאות לכך הם sin(x)/x ו- exp(-x2). אגב, עם הבעיה הזו אפשר להתמודד גם בעזרת תור טיילור.
* לא תמיד f(x) ידועה בצורה אנליטית, אלא כאוסף של נקודות ידועות או כתוצאה של חישוב עלום.

התחום של אינטגרציה נומרית הוא מאד רחב ואנחנו נכנס רק לשיטות הבסיסיות ביותר.

השיטות הבסיסיות הם די פשוטות. הם מבוססות על חישוב מצולעים שמקרבים את שטח האינטגרל, המתקבלים ע"י חלוקת הקטע (a,b) לקטעים קטנים.

שיטה בסיסית אחת היא שיטת הטרפז:



שטח הטרפז ש"מקרבת" את השטח הוא

***I(a,b) ≈ (b – a)(f(a) – f(b))/2***

הסכמה של חישוב I(a,b) הינו לחלק את הקטע ל-n יחידות בגודל h= (b-a)/n. ככל נבחר n גדול יותר נקבל h קטן יותר הנותן דיוק תאורטי יותר גדול אך מגדיל את מספר החישובים ובנוסף לכך ככל ש-h קטן יותר גדל הסכנה של שגיאות עיגול שיכולים לשבש את הדיוק של התוצאה.

נוסחת הקירוב המתקבלת היא

***h=(b-a)/n***

***I(a,b) ≈ T(a,b,h) = h((f(a) + f(b))/2 + Σnk=1 f(a +kh))***

**קוד C**

long double trapezoid(long double (\*f)(long double),

 long double a, long double b, int n)

{

 long double h, I, t;

 int i;

 h = (b - a)/n;

 I = ((\*f)(a)+ (\*f)(b))/2.0;

 t = a;

 for(i=1; i < n; i++)

 {

 t = t + h;

 I = I + (\*f)(t);

 } /\* for \*/

 I = I\*h;

 return I;

} /\* trapezoid \*/

**שיטת המלבן או נקודת האמצע**

 שיטה דומה לשיטת הטרפז היא שיטת המלבן. הנוסחה הבסיסית שהיא מבוסס עליה היא

***I(a,b) ≈ (b – a)(f((a+b)/2))***

כאשר אנחנו בוחרים לקרב את האינטגרל אנחנו מקבלים את הנוסחה

***h = (b-a)/n***

***I(a,b) ≈ M(a,b,h) = h(Σnk=1 f(a +(k+0.5)h))***

**קוד C**

long double midpoint(long double (\*f)(long double),

 long double a, long double b, int n)

{

 long double h, I, t;

 int i;

 h = (b - a)/n;

 I = 0;

 t = a+h/2.0;

 for(i=0; i < n; i++)

 {

 I = I + (\*f)(t);

 t = t + h;

 } /\* for \*/

 I = I\*h;

 return I;

} /\* midpoint \*/

**חישוב n או h מספק**

 כאמור בחירת n גדול או (בצורה שקולה h קטן) מגדיל את הדיוק התאורטי אבל מגדיל את מספר החישובים ואת הסכנה לשגיאות הנובעות משגיאות עיגול. אפשר להתמודד עם הבעיה בצורה דומה לחישוב קירוב הנגזרת. להתחיל מ-n קטן (או h גדול) להקטין אותו עד שנראה שהקטנות נוספות כבר אינן משנות הרבה את הפתרון למשל אם

 ***| T(a,b,h) - T(a,b,h/2) | < ε***

להלן סכמה של מציאת n מתאים בקוד C:

long double trapez\_schema(long double (\*f)(long double),

 long double a, long double b, long double eps)

{

 int n;

 long double t1, t2;

 n = 128;

 t2 = trapezoid(f, a, b, n);

 do {

 n = n \* 2;

 t1 = t2;

 t2 = trapezoid(f, a, b, n);

 } while(fabsl(t1 - t2) > eps);

 return t2;

} /\* trapez\_schema \*/

קריאה אפשרית לסכמה הינה

I = trapez\_schema(sqr, 0, 2.0,0.00000000001);

צורה מעניינת נוספת להתמודד עם הבעיה הוא לשלב בין שיטת הטרפז והמלבן.

שתי השיטות מקיימות את הנוסחה הבאה:

***T(a,b,h/2) = (T(a,b,h) +M(a,b,h))/2***

הוכחה:

***T(a,b,h) +M(a,b,h) =***

***h((f(a) + f(b))/2 + Σnk=1 f(a +kh)) + h(Σnk=1 f(a +(k+0.5)h)) =***

***h(f(a) + f(b))/2 + h Σnk=1 f(a +kh)) + h(Σnk=1 f(a +(k+0.5)h)) =***

***h(f(a) + f(b))/2 + h Σ2nk=1 f(a +kh/2))***

לפיכך

***(T(a,b,h) +M(a,b,h))/2 =***

 ***(h/2)(f(a) + f(b))/2 + (h/2) Σ2nk=1 f(a +kh/2)) = T(a,b,h/2)***

מש"ל.

**קוד C של השילוב**

long double combi(long double (\*f)(long double),

 long double a, long double b,

 long double eps)

{

 long double h, t, T,M;

 int i, k, n;

 h = b - a;

 n = 1;

 T = h\*((\*f)(a) + (\*f)(b))/2.0;

 for(k=0; k < 30; k++)

 {

 M=0;

 t = a+0.5\*h;

 for(i=0; i < n; i++)

 {

 M = M + (\*f)(t);

 t = t + h;

 } /\* for \*/

 M = M\*h;

 T = (T+M)/2.0;

 if ( fabsl(T-M) < eps )

 return T;

 h = h/2.0;

 n = n\*2;

 } /\* for \*/

 return T;

} /\* combi \*/

**שיטת simpson לאינטגרציה נומרית**

 שיטת סימפסון מחשבת את הקירוב ע"י ביצוע אינטגרציה על בכל קטע h על אינטרפולציה של הפונקציה ע"י פונקציה ריבועית. האינטרפולציה הזו נקבעת ע"י ערכי הפונקציה המקורית בקצות הקטע ובנקודת האמצע.

גרפית הקירוב יכול להראות כך:



את נוסחת האינטרפולציה אפשר לקבל למשל מנוסחת לגרנז:

יהיו a,b הקצוות של הקטע ו-m = (a+b)/2.

***P(x) = f(a)(x-m)(x-b)/((a-m)(a-b)) +***

 ***f(m)(x-a)(x-b)/((m-a)(m-b)) +***

 ***f(b)(x-a)(x-m)/((b-a)(b-m))***

אם היינו עושים את האלגברה היינו מגיעים לכך שנוסחת האינטגרל היא

***I(a,b) ≈ ((b-a)/6)(f(a) +4f((a+b)/2) +f(b))***

אם נחלק את הקטע ל-n קטעים בגודל h = (b-a)/n, נעבור את האינטרפולציה ונעשה אינטגרציה עליו נקבל

***xi = a + ih, i = 0,1, … , b***

***I(a,b) ≈ (h/3)( f(x0) + 4 f(x1) +2 f(x2) + 4 f(x3) + … + f(xn) )***

או

***I(a,b) ≈ (h/3)\*( f(x0) + 2 Σn/2-1j=1f(x2j) + 4 Σn/2j=1f(x2j-1) + f(xn) )***

**קוד C**

long double simpson(long double (\*f)(long double),

 long double a, long double b, int n)

{

 long double h, I, t, fact;

 int i;

 h = (b - a)/n;

 I = (\*f)(a)+ (\*f)(b);

 t = a;

 fact = 4.0;

 for(i=1; i < n; i++)

 {

 t = t + h;

 I = I + fact\*(\*f)(t);

 if (fact == 4.0)

 fact = 2.0;

 else

 fact = 4.0;

 } /\* for \*/

 I = I\*h/3;

 return I;

} /\* simpson \*/