Numerical Methods II

Project 1

Task 24: Numerical calculation of the integral $\int_{-1}^{1} p(x)dx$, where $p(x) = a_0 T_0(x) + a_1 T_1(x) + \ldots + a_k T_k(x)$ is a polynomial given in the basis of Chebyshev polynomials of the first kind. Use the n-point ($n = 3, 5, 7, \ldots, 15$) Gauss-Legendre rule.

November 9, 2018

Rafał Ziomek

Group: EA2

Contents

Method description 2
- Gauss-Legendre quadrature rule 2
- Legendre polynomials 2
- Chebyshev polynomials 2
- Calculation steps 3

Matlab functions code 4
- legendre 4
- getlegendreroots 5
- integrweights 6
- chebpolyval 7
- integratechebyshev 8
- Testing functions 9
  - getcoefs 9
  - getchebcoefs 10
  - checkchebintegr 10
  - test 11

Numerical examples and analysis 13

Method description

Gauss-Legendre quadrature rule

Method I will use in my calculations is n-point Gauss-Legendre quadrature rule. It is constructed to yield an exact result for polynomials of degree $2n-1$ or less by a suitable choice of the nodes x_i and weights w_i for $i = 1, 2, ..., n$. The rule in the domain $[-1, 1]$ is stated as:

$$\int_{-1}^{1} f(x)dx \approx \sum_{i=1}^{n} w_i f(x_i)$$

which is exact for polynomials of degree $2n-1$ or less. The i-th node x_i is the i-th root of Legendre polynomial P_n and corresponding weights are given by the formula:

$$w_i = \frac{2}{(1-x_i^2)[P_n'(x_i)]^2}$$

Legendre polynomials

Legendre polynomials P_n of degree n are defined explicitly as:

$$P_n(x) = \frac{1}{2^n} \sum_{k=0}^{\lfloor \frac{n}{2} \rfloor} (-1)^k \binom{n}{k} \binom{2n-2k}{n} x^{n-2k}$$

Chebyshev polynomials

Chebyshev polynomials of the first kind T_n of degree n are defined recursively as follows:

$$T_0(x) = 1$$

$$T_1(x) = x$$

$$T_{n+1}(x) = 2xT_n(x) - T_{n-1}(x)$$

and explicitly as:

$$T_n(x) = \frac{n}{2}\sum_{k=0}^{\lfloor \frac{n}{2} \rfloor}(-1)^k \frac{(n-k-1)!}{k!(n-2k)!}(2x)^{n-2k}$$

Calculation steps

In order to apply Gauss-Legendre method, the program does following steps:

(S_1) Calculating coefficients of Legendre polynomial P_n (n nodes in method).

(S_2) Calculating n roots of P_n.

(S_3) Calculating n corresponding weights.

(S_4) Calculating $p(x_i) = a_0T_0(x_i) + a_1T_1(x_i) + ... + a_kT_k(x_)$ for all roots from S_2

(S_5) Calculating the integral $\int_{-1}^{1} p(x)dx$

MATLAB FUNCTIONS CODE

legendre

```
%Function used for calculations of Legendre polynomial of degree n
%Usage: legendre(n), n is desired degree
%Output: coefficients vector
function [N] = legendre(n)
N0=1;
N1=1;
if n<0
    N=-1;
    disp("legendre(n), n must be greater or equal 0.")
    return;
end
if n==0
    N=N0;
    return;
end
if n==1
    N=[N0, N1];
    return;
end

N = zeros(1, n+1);
k = n/2;
index = 1;
%Implemenation of explicit formula for Legendre polynomial
for i=0 : k
    N(index) = 1/2^n * (-1)^i * nchoosek(n, i) * nchoosek(2*n-2*i, n);
    index = index + 2;
end

return;
```

getlegendreroots

```
%Function calculating roots of given legendre polynomial using built−in
%MatLAB function
%Usage: getlegenferroots(l), where l is coefficeints vector
%Output: roots vector
function [x] = getlegendreroots(l)

if l<1
    x=NA;
    disp("getlegendreroots(n), n must be greater than 0")
    return;
end

if l==1
    x=0;
    return;
end

x = roots(l);
return
```

integrweights

```
%Function calculating integration weights
%Usage integrweights(x, legendre), x - roots vector, legendre -
%coefficients vector
%Output: weights vector
function [w] = integrweights(x, legendre)

len = length(x);
dl = polyder(legendre); %first derivative of given polynomial
w = zeros(1, len);

for i=1:len
    w(i) = 2/((1-x(i)^2)*polyval(dl,x(i))^2);
end
return
```

chebpolyval

```
%Function calculating value of p(x)=a0T0(x)+a1T1(x)+...+anTn(x) using
    recursive
%relation of Chebyshev polynomials
%Usage: chebpolyval(a, x), a - coefficients vector, x - variable
%Output: value of p(x)
function [t] = chebpolyval(a, x)
n = length(a)-1;

if n == 0
    t = a;
    return
end

if n == 1
    t = a(2)*x + 1;
    return
end

t0 = 1;
t1 = x;
tx = 2*x;
t = t0*a(1)+t1*a(2);

for i=2:n
    t2 = tx*t1 - t0;
    t0 = t1;
    t1 = t2;
    t = t + a(i+1)*t2;
end

return
```

integratechebyshev

```
%Desired function calculating integral by Gauss_Legendre method
%Usage: integratechebyshev(a, n), a-coefficients vector, n-desired number
%of integration points
%Output: value of the integral

function [y] = integratechebyshev(a, n)

lcoef = legendre(n);
x = getlegendreroots(lcoef);
w = integrweights(x, lcoef);
y = 0;

for k=1:n
    y = y + w(k)*chebpolyval(a, x(k));
end
return
```

Testing functions

getcoefs

```
%Function calculating coefficients of Chebyshev polynomial of degree n
%using explicit formulation, used in tests
%Usage: getcoefs(n), n - degree of polynomial
%Output: coefficients vector

function [p] = getcoefs(n)
if n<0
    p = NA;
    return
end

if n == 0
    p = 1;
    return
end

if n == 1
    p = [1,0];
    return
end

p = zeros(1, n+1);
k = n/2;
index = 1;
for i=0 : k
    p(index) = k*(-1)^i*(factorial(n-i-1)/(factorial(i)*factorial(n-2*i
        )))*2^(n-2*i);
    index = index + 2;
end
return
```

getchebcoefs

```
%Function calculating coefficients of p(x)=a0T0(x)+a1T1(x)+...+anTn(x),
%used in tests
%Usage: getchebcoefs(a), a-vector of coefficients
%Output: vector of coefficients of p(x)
function [p] = getchebcoefs(a)

n = length(a)-1;
p = getcoefs(n);
p = p .* a(n+1);

for i=1 : n
    z = zeros(1,i);
    c = getcoefs(n-i);
    c = c .* a(n-i+1);
    p = p + [z,c];
end

return
```

checkchebintegr

```
%Function for testing, calulating integral using all coefficients and
%built-in MatLAB functions
%Usage: checkchebintegr(a), a - vector of coefficients
%Output: value of p(x) integral
function [y] = checkchebintegr(a)

t = getchebcoefs(a);
y = diff(polyval(polyint(t), [-1 1]));

return
```

test

```
%Tests of Gauss_Legendre method for calculating p(x). Tests consist of
%integration of polynomial of smaller than 2n-1, equal 2n-1 and equal to
%2n, where n is number of points of integration. As expected, method works
%for polynomials of degree smaller or equal 2n-1 and produces significant
%error for polynomials of higher degree

a0=rand(1,3) %3 random coefficients, degree of polynomial=2
calculated_value=integratechebyshev(a1,3) %value calculated by implemented method, 3 integration points
actual_value=checkchebintegr(a1) %actual value
error=actual_value-calculated_value %error

a1=rand(1,6) %6 random coefficients, degree of polynomial=5
calculated_value=integratechebyshev(a1,3) %value calculated by implemented method, 3 integration points
actual_value=checkchebintegr(a1) %actual value
error=actual_value-calculated_value %error

a2=rand(1,7) %7 random coefficients, degree of polynomial=6
calculated_value=integratechebyshev(a2,3) %value calculated by implemented method, 3 integration points
actual_value=checkchebintegr(a2) %actual value
error=actual_value-calculated_value %error

a3=rand(1,12) %12 random coefficients, degree of polynomial=11
calculated_value=integratechebyshev(a3,9) %value calculated by implemented method, 9 integration points
actual_value=checkchebintegr(a3) %actual value
error=actual_value-calculated_value

```

```
a4=rand(1,18) %18 random coefficients, degree of polynomial=17
calculated_value=integratechebyshev(a4,9) %value calculated by
    implemented method, 9 integration points
actual_value=checkchebintegr(a4) %actual value
error=actual_value-calculated_value %error

a5=rand(1,19) %19 random coefficients, degree of polynomial=18
calculated_value=integratechebyshev(a5,9) %value calculated by
    implemented method, 9 integration points
actual_value=checkchebintegr(a5) %actual value
error=actual_value-calculated_value %error

a6=rand(1,20) %20 random coefficients, degree of polynomial=19
calculated_value=integratechebyshev(a6,15) %value calculated by
    implemented method, 15 integration points
actual_value=checkchebintegr(a6) %actual value
error=actual_value-calculated_value %error

a7=rand(1,30) %30 random coefficients, degree of polynomial=29
calculated_value=integratechebyshev(a7,15)%value calculated by
    implemented method, 15 integration points
actual_value=checkchebintegr(a7) %actual value
error=actual_value-calculated_value %error

a8=rand(1,31) %31 random coefficients, degree of polynomial=30
calculated_value=integratechebyshev(a8,15) %value calculated by
    implemented method, 15 integration points
actual_value=checkchebintegr(a8) %actual value
error=actual_value-calculated_value %error
```

NUMERICAL EXAMPLES AND ANALYSIS

Running test from previous section gives following errors:

1. $deg = 2 \; n = 3$
$$err = 1.110223024625157e-16$$
2. $deg = 5 \; n = 3$
$$err = -1.110223024625157e-16$$
3. $deg = 6 \; n = 3$
$$err = 0.716454475956586$$
4. $deg = 11 \; n = 9$
$$err = 3.463895836830488e-14$$
5. $deg = 17 \; n = 9$
$$err = -5.591305196617213e-12$$
6. $deg = 18 \; n = 9$
$$err = 1.488299646766813$$
7. $deg = 19 \; n = 15$
$$err = -2.265296283887608e-11$$
8. $deg = 29 \; n = 15$
$$err = -9.492332031513229e-09$$
9. $deg = 30 \; n = 15$
$$err = 1.106201089446377$$

As expected, method is working accurately for polynomials of degree greater or equal to $2n-1$. When polynomial is of higher degree, the error is significant.