Numerisk analys, MA1454 13 augusti 2016

Laboration 2 MA1454

Problem 7

Oliver Rosander, 940314-3739

Sammanfattning

Inledning

I denna laborationsrapport kommer automatisk integration med Monte Carlo-metoden utföras med hjälp av ekvationslösning, differensapproximation och interpolering.

Deluppgift (a)

Problem

Givet ett polynom $p(x)$ ska lösningar beräknas då $p(x) = 0$. Genom att manuellt dela upp intervallet $I = [a, b]$ i delintervall av längden h som bör vara större än antalet uppskattade lösningar till $p(x) = 0$ i intervallet. På så sätt går det att avgöra om I innehåller en lösning till $p(x) = 0$. Dessa approximativa lösningar $(x_0, x_1, ..., x_{n-1}, x_n)$ sådana att $p(x_k) = 0$ ska returneras i en vektor där $x_{k-1} < x_k$ måste vara uppfyllt.

Lösning

En storlek på intervallen bestäms manuellt av användaren för att anpassa storleken på delintervallen till det polynom vars integral ska beräknas. Så länge intervallen är mindre eller lika med slutpunkten undersöks det om det finns en lösning till $f(x) = 0$ genom att beräkna $f(x_{k-1}$ och $f(x_k)$. Är produkten av dessa två tal negativt finns det en lösning till f(x) i intevallet $[x_{k-1}, x_k]$ enligt Bolzanos sats [1].

En mittpunkt beräknas som används för att bestämma i vilket delintervall lösningen finns i så det går att halvera intervallen mellan x_{k-1} och x_k. Detta görs till $xB - xA > precision$ där *precision* variabeln bestäms utav användaren i lab2.m filen. Då har en lösning approximativt funnits i m med säkerheten som användaren har bestämt.

När alla intervall har genomsökt för lösningar returneras dess lösningar i en vektor.

lab2_solutions.m

```
function [retVector] = lab2_solutions(f, a, b)
format long
    h = (b-a) / 17;
    subIntervalA = a;
    subIntervalB = a + h;
    retVector = [];

    while(subIntervalB <= b)
        yA = f(subIntervalA);
        yB = f(subIntervalB);

        res = yB * yA;
        if(res < 0)
            xA = subIntervalA;
```

```
            xB = subIntervalB;
            if xB-xA > eps
                while xB-xA > (eps+20*10^-16)
                    m = (xA+xB)/2;
                    if f(xA)*f(m) < 0
                        xB = m;
                    elseif f(xB)*f(m) < 03
                        xA = m;
                    elseif f(xA)*f(m) < 0 || f(xB)*f(m) < 0
                        xA = m;
                        xB = m;
                    end
                end
                retVector = [retVector, m];
            end
        end
        subIntervalA = subIntervalB;
        subIntervalB = subIntervalB + h;
    end
    retVector = unique(retVector);
end
```

Deluppgift (b)

Problem

Med hjälp av de delintervall $I_k = [x_{k-1}, x_k]$ där $k = 1, 2, ..., n$ som är lösningar till polynomet $f(x_k) = 0$ ska extremvärden beräknas för varje intervall. Genom att approximera $f'(m)$ i $m + 1$ ekvidistanta punkter med central differensapproximation och sedan interpolera $f'(x)$ i intervallet I_k med ett polynom $p_m(x)$ kan extremvärdena beräknas då $p_m(x) = 0$. Dessa lösningar returneras i en vektor.

Lösning

Genom att loopa över alla intervall I_k och estimera m+1 värden till $f'(x)$ med differensapproximation går det att interpolera de punkterna med ett polynom som används för att finna extremvärdena i intervallet I_k [1].

Finns det ingen lösning till $f'(x) = 0$ vilket uppstår i det första och sista intervallet eller om $f(x) = 0$ saknar en lösning i intervallet $[a, b]$. Då sätts extremvärdena till $f(a)$ för det första intervallet om $|f(x_0)| < |f(x_1)|$ och det sista extremvärdet sätts till $f(b)$ då det alltid är ett extremvärde i det sista intervallet givet att antal nollpunkter är större än noll. Finns det inga nollpunkter sätts extremvärdet till den punkt med störst absolutvärde.

lab2_localExtreme.m

```
function [retVector] = lab2_localExtreme(inVector, f, a, b)
  %Add a and b to have the complete intervalls
  m = 5;
  inVector = [a, inVector];
```

```
  inVector = [inVector, b];
  yValues = [];
  %loop over intervals
  for i=2:length(inVector)
      dfValues = [];
      xValues = [];
      I = [inVector(i-1)];
      I = [I, inVector(i)];

      %split intervall
      h = (I(2)-I(1)) / (m+1);
      next = I(1) + h;
      %calculate df
      while next < I(2)
         temp = vpa((-f(next+2*h) + 8*f(next+h) - 8*f(next-h) + f(next+2*h)) / 12*h);
         dfValues = [dfValues, temp];
         xValues = [xValues, next];
         next = next + h;
      end
      %interpolation
      Vx = vander(xValues);
      grid on
      hold on
      syms x;
      coefficents = double(Vx \ dfValues');
      derivativeF(x) = poly2sym(coefficents);
      x = linspace(I(1), I(2));
      plot(x, [derivativeF(x); f(x)])
      %find solutions to f'(x) = 0
      solution = lab2_solutions(derivativeF, I(1), I(2));
      if (isempty(solution) && i == 2) && abs(f(I(1))) > abs(f(I(2))
          solution = I(1);
      elseif isempty(solution)
          solution = I(2);
      end
      yValues = [yValues, f(solution)];
  end
  retVector = vpa(yValues);
d
```

Deluppgift (c)

Problem

Givet vektorn med lösningar till $p(x) = 0$ och vektorn med extremvärdena i intervallen $I_k = [x_k, x_{k-1}]$ där $k = 1, 2, ..., n$ kan integalen $\int_{x_{k-1}}^{x_k} p(x)dx$. Genom att summera dessa integraler för alla intervall I_k kan $\int_a^b f(x)dx$ upskattas.

Lösning

Med ett polynom, ett intervall $[a, b]$ och en extrempunkt beräknas integralen genom att slumpa fram 1000 x och y värden som används för att beräkna och jämföra om $f(x) > y$ och räkna hur många sådana punkter som finns. Genom att beräkna hur stor procentuell andel av punkterna som är under polynomet kan en sådan area uppskattas.

lab2_integrate.m

```
function [retValue] = lab2_integrate(f, a, b, c)
  negative = 0;
  less = 0;
  numberOfDots = 1000;
  xRandVector = (b-a).*rand(numberOfDots,1) + a;
  yRandVector = c.*rand(numberOfDots,1);


  if c < 0
      negative = 1;
  end
  for i=1:numberOfDots
      if negative == 0 && f(xRandVector(i)) >= yRandVector(i)
          less = less + 1;
      elseif negative == 1 && f(xRandVector(i)) <= yRandVector(i)
          less = less + 1;
      end
  end
  area = (b-a)*c*less / numberOfDots;
  retValue = double(area);
end
```

Genom att summera alla dessa integraler kan den slutgiltiga integralen $\int_a^b f(x)dx$ estimeras.

```
for i = 2:length(zero)
   tic
   start = zero(i-1);
   stop = zero(i);
   extremeValue = extreme(i-1);
   integral = integral + lab2_integrate(f, start, stop, extremeValue);
   toc
end

integral
```

Deluppgift (d)

Programmet ska testas med fyra olika polynom i olika intervaller. De fyra polynomen som valt ut är av olika karaktärer. Efter som Monte Carlo metoden är en probabalistisk beräkning kommer resultatet variera för varje körning programmet körs. Det går att förbättre resultatet genom att generera fler punkter men då tar beräkningen dessutom längre tid.

- $\int_0^6 0.5xcos(2x)dx = -0.7702.$
- $\int_{-3}^3 0.9x^3 - 4x + 1dx = 5.7807.$
- $\int_{-5}^5 -xdx = 0.1500.$
- $\int_{-2}^2 x^2 - 1dx = 1.2680.$

Programmet som används för att koppla samman alla delar och summera integralerna är infogat nedan.

lab2.m

```
close all
clear all
hold on

syms x
%f(x) = 0.5*x*cos(2*x); a = 0; b = 6;
%f(x) = 0.9*x^3-4*x+1; a = -3; b = 3;
%f(x) = -x; a = -5; b = 5;
f(x) = x^2-1; a = -2; b = 2;

integral = 0;
x = linspace(a,b);


zero = [];
zero = lab2_solutions(f, a, b);


extreme = lab2_localExtreme(zero, f, a, b);
zero = [a, zero, b];

for i = 2:length(zero)
   tic
   start = zero(i-1);
   stop = zero(i);
   extremeValue = extreme(i-1);
   integral = integral + lab2_integrate(f, start, stop, extremeValue);
   toc
end

integral
```

Referenser

[1] R.Nyqvist. (2016, 18 Maj). Numerisk Analys Föreläsningsanteckningar (Version 4.66920160910299). Online. Tillgänglig: bth.itslearning.com.