Thursday, 11/14/13 Math 471, Fall 2013, Section 001 Topic 7

1 Numerical differentiation

Problem : Given a numerical representation of a function's data, how do we approximate that function's derivative?

- Finite difference methods
  - Related : finite volume methods
- Find an interpolating or approximating function, take that function's derivative
  - Related : finite element methods
- Use Fourier series or transforms: differentiation becomes simple multiplication
  - Related : spectral methods
- Use Green's functions and integral kernels to transform differentiation into an equivalent integration problem
  - Related : vortex / particle methods

2 Numerical integration

Problem : Given an numerical representation of a function's data, how can we approximate that function's definite integral?

2.1 Introduction

In one space dimension, the problem is more precisely stated as: can we find coefficients c_i such that

$$\int_a^b f(x)\,dx \approx \sum_{i=1}^{n} c_i f(x_i) \quad ?$$

Riemann sums:

Given an interval $a = x_0 < x_1 < \cdots < x_n < x_{n+1} = b$ with $n+2$ points, $n+1$ subdivisions, and data $f_i = f(x_i),\ i = 0 : n+1$:

Left-hand sum:

$$\int_a^b f(x)\,dx \approx \sum_{n=0}^{n} f(x_i)(x_{i+1} - x_i)$$

Right-hand sum:

$$\int_a^b f(x)\,dx \approx \sum_{n=1}^{n+1} f(x_i)(x_i - x_{i-1})$$

Trapezoid rule :

$$\int_a^b f(x)\,dx \approx \sum_{n=0}^{n} \left(\frac{f(x_{i+1}) - f(x_i)}{2} \right) (x_{i+1} - x_i)$$

This should all be review; but let's take a more detailed look at the trapezoid rule.
For now, let's assume we have a uniform mesh indexed as we did in the interpolation chapter, that is,

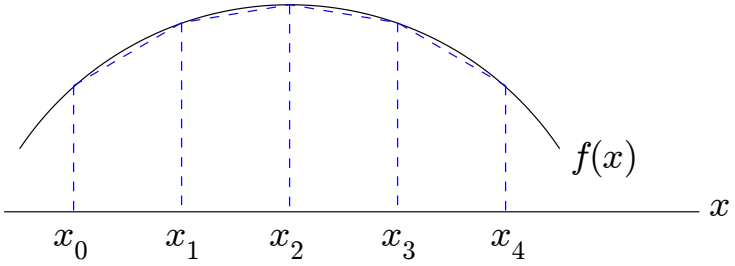
$$a = x_0 < x_1 < \cdots < x_{n-1} < x_n = b$$

for total of $n+1$ points and n subdivisions with mesh size $h = \frac{b-a}{n}$. For such a mesh, we may write the trapezoid rule as a function of the mesh size h:

$$T(h) = h \cdot \frac{1}{2}[f(x_0) + f(x_1)] + h \cdot \frac{1}{2}[f(x_1) + f(x_2)] + \cdots + h \cdot \frac{1}{2}[f(x_{n-1}) + f(x_n)]$$

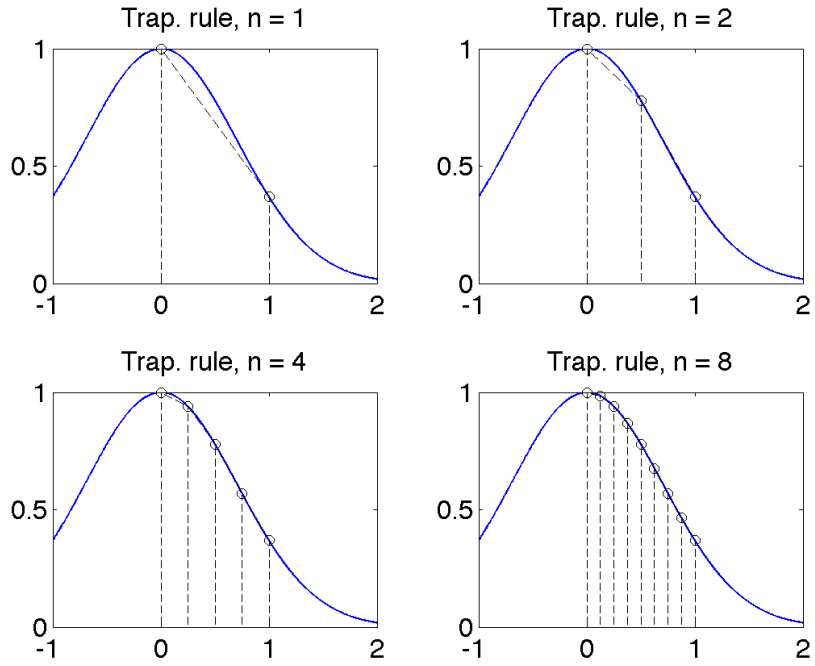
$$T(h) = h \cdot [\frac{1}{2}f(x_0) + f(x_1) + \cdots + f(x_{n-1}) + \frac{1}{2}f(x_n)]$$

Graphically, the trapezoid finds the sum of the areas of the trapezoids formed by this mesh underneath the curve $f(x)$:



Note : The trapezoid rule computes the area underneath the piecewise linear interpolant to $f(x)$, even for nonuniform meshes.

Example 1: $\int_0^1 e^{-x^2}\,dx = 0.746824132812427\ldots$



n	h	$T(h)$	error	error$/h^2$
1	1	0.683939720585721	0.062884412226706	0.062884412226706
2	0.5	0.731370251828563	0.015453880983864	0.061815523935456
4	0.25	0.742984097800381	0.003840035012046	0.061440560192732
8	0.125	0.745865614845695	0.000958517966732	0.061345149870832

The table suggests that the trapezoid rule is second order accurate; let's prove it.

△

Theorem 1. *For a given interval $a \le x \le b$ divided uniformly into n subdivisions by $n+1$ points with $h = (b-a)/n$ and function data $f_i = f(x_i)$ defined on the points, the trapezoid rule*

$$T(h) = h\left(\frac{1}{2}f(x_0) + f(x_1) + \cdots + f(x_{n-1}) + \frac{1}{2}f(x_n)\right)$$

satisfies

$$\int_a^b f(x)\,dx = T(h) - \frac{1}{12}(b-a)f''(\xi)h^2$$

for some $\xi \in [a,b]$.

Proof.

local error:

For $x \in [x_i, x_{i+1}]$, p/w linear interp. $\Rightarrow f(x) = q_i(x) + \frac{1}{2}f''(\xi)\cdot(x-x_i)(x-x_{i+1})$ for some $\xi \in [x_i, x_{i+1}]$

Recall : $q_i(x) = f[x_i] + f[x_i, x_{i+1}](x - x_i)$, $\qquad x_{i+1} - x_i = h$

$$\int_{x_i}^{x_{i+1}} f(x)\,dx = \int_{x_i}^{x_{i+1}} f[x_i] + f[x_i, x_{i+1}](x - x_i) + \underbrace{\frac{1}{2} f''(\xi)(x - x_i)(x - x_{i+1})}_{\text{integrate by parts}}\,dx$$

$$= hf[x_i] + f[x_i, x_{i+1}]\frac{h^2}{2} - \frac{1}{12} f''(\xi)h^3$$

$$\int_{x_i}^{x_{i+1}} f(x)\,dx = \underbrace{h \cdot \frac{1}{2}\left(f(x_i) + f(x_{i+1})\right)}_{\text{trap. rule}} - \underbrace{\frac{1}{12} f''(\xi)h^3}_{\text{local trunc. err.}}$$

global error:

$$\int_a^b f(x)\,dx = \sum_{i=0}^{n-1} \int_{x_i}^{x_{i+1}} f(x)\,dx$$

$$= \sum_{i=0}^{n-1} \left(h \cdot \frac{1}{2}\left((f(x_i) + f(x_{i+1})\right) - \frac{1}{12} f''(\xi_i)h^3 \right)$$

$$= T(h) - \frac{1}{12} n \cdot f''(\xi)h^3 = T(h) - \frac{1}{12}(b - a) f''(\xi)h^2 : \text{second order accuracy}$$

□

Question 1: How could we get better accuracy?

- Midpoint rule, $M(h)$: $\int_a^b f(x)\,dx = M(h) - \frac{b-a}{24} f''(\xi)h^2$

$$M(h) = h \cdot \left[f\left(\frac{x_0 + x_1}{2}\right) + \cdots + f\left(\frac{x_{n-1} + x_n}{2}\right) \right]$$

- integrate a piecewise quadratic interpolant (Simpson's rule)
- integrate a cubic spline interpolant
- non-uniform (adaptive) meshes
- asymptotic expansion

△

2.2 Richardson extrapolation (Romberg's method)

Text : section 6.7

Write the trapezoid rule as an asymptotic series in powers of h, valid in the limit $h \to 0$:

$T(h) = \int_a^b f(x)\,dx + c_2 h^2 + c_4 h^4 + c_6 h^6 + \cdots$, where the coefficients c_i are independent of h

Then

$T(2h) = \int_a^b f(x)\,dx + c_2 (2h)^2 + c_4 (2h)^4 + c_6 (2h)^6 + \cdots$

$$= \int_a^b f(x)\,dx + 4c_2h^2 + 16c_4h^4 + 64c_6h^6 + \cdots$$

$$\Rightarrow \frac{4T(h) - T(2h)}{3} = \frac{1}{3}\left(3\int_a^b f(x)\,dx - 12c_4h^4 - 60c_6h^6 + \cdots\right)$$

$$= \int_a^b f(x)\,dx - 4c_4h^4 - 20c_6h^6 + \cdots$$

Define $R_0(h) = T(h)$: second order accurate

$$R_1(h) = R_0(h) + \frac{R_0(h) - R_0(2h)}{3} \text{ : 4th order accurate}$$

$$R_1(h) = \int_a^b f(x)\,dx + \tilde{c}_4h^4 + \tilde{c}_6h^6 + \cdots$$

$$R_1(2h) = \int_a^b f(x)\,dx + \tilde{c}_4(2h)^4 + \tilde{c}_6(2h)^6 + \cdots$$

$$= \int_a^b f(x)\,dx + 16\tilde{c}_4h^4 + 64\tilde{c}_6h^6 + \cdots$$

$$\Rightarrow \frac{16R_1(h) - R_1(2h)}{15} = \int_a^b f(x)\,dx - \frac{16}{5}\tilde{c}_6h^6 + \cdots$$

Define $R_2(h) = R_1(h) + \dfrac{R_1(h) - R_1(2h)}{15}$: 6th order accurate

Keep it going...
<u>difference table</u>

$$\begin{array}{lllll}
2h & R_0(2h) & & & \\
 & & \frac{\Delta}{3} & & \\
h & R_0(h) & & R_1(h) & \\
 & & \frac{\Delta}{3} & & \frac{\Delta}{15} \\
\frac{1}{2}h & R_0(\frac{1}{2}h) & & R_1(\frac{1}{2}h) & & R_2(\frac{1}{2}h)
\end{array}$$

<u>Example 2:</u> $\int_0^1 e^{-x^2}\,dx = 0.746824132812427\ldots$

Set $h = 0.5$, so that $2h = 1$, $\frac{h}{2} = 0.25$, etc...

h	$R_0(x)$	$R_1(x)$	$R_2(x)$	$R_3(x)$
1	0.683939720585721			
0.5	0.731370251828563	0.747180428909510		
0.25	0.742984097800381	0.746855379790987	0.746833709849752	
0.125	0.745865614845695	0.746826120527467	0.746824169909899	0.746824018482282

method	error
$T(0.125)$	9.585179667317423e-04
$R_3(x)$	1.143301452399825e-07

Notes:

- The last column $R_3(h)$ uses $\dfrac{\Delta}{63}$
- down a column : decreasing h, fixed order of accuracy
- across a row : fixed h, increased order of accuracy

$\triangle$

2.3 Adaptive quadrature

Text : section 6.8

Example 3: Consider approximations of the integral $\displaystyle\int_0^1 x^{16}\cos(x^{16})\,dx$.

The second derivative of the integrand is

$$f''(x) = 240x^{14}\cos(x^{16}) - 512x^{30}\sin(x^{16}) + x^{16}(-256x^{30}\cos(x^{16}) - 240x^{14}\sin(x^{16}));$$

it is bounded on $[0,1]$ by $240 + 512 + 256 + 240 = 1248$,

$$\max_{0\le x\le 1} \left|f''(x)\right| \le 1248,$$

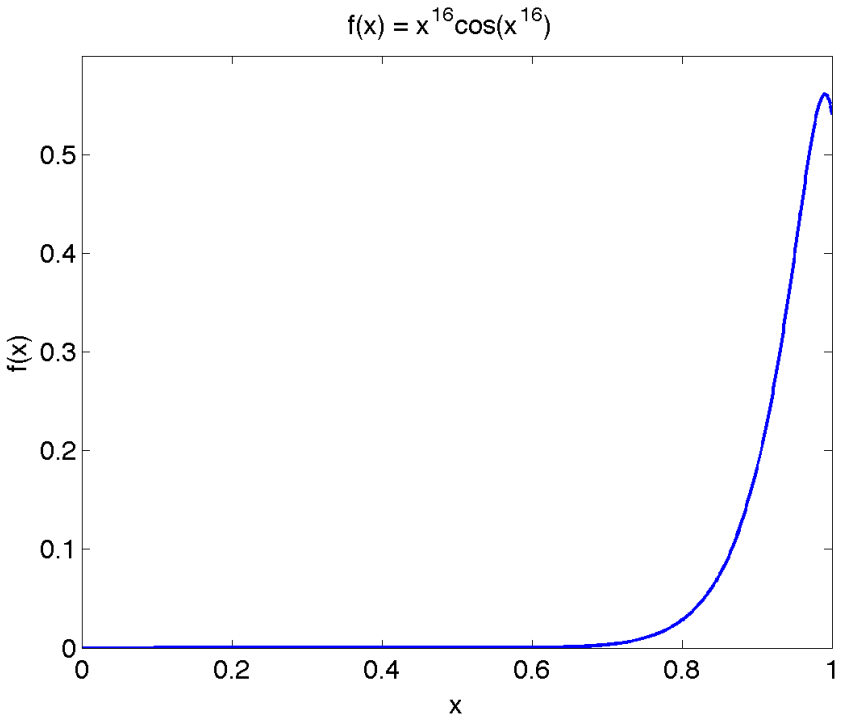
which means errors in the midpoint rule will satisfy

$$\left|\int_0^1 x^{16}\cos(x^{16})\,dx - M(h)\right| \le 52h^2.$$

Using a uniform mesh for [0,1], we have $h = 1/n$ and an error tolerance of ϵ_{tol} would require

$$52h^2 \le \epsilon_{\text{tol}} \quad \Rightarrow \quad \frac{52}{n^2} \le \epsilon_{\text{tol}} \quad \Rightarrow \quad n \ge \left(\frac{52}{\epsilon_{\text{tol}}}\right)^{1/2}$$

Thus for a tolerance of 1×10^{-6}, we would need to use $n \ge 7211$ to guarantee an accurate result.
A plot of the integrand quickly shows why this is inefficient –



– the integrand is nearly constant for the majority of the interval; the area under that part of the curve could be accurately calculated with a coarse mesh.

△

2.3.1 adaptive strategy 1: grid mapping functions

Suppose we know in advance that one part of the domain will have large variations in the integrand, but other parts of the domain will not (for example, we may know in advance that a problem contains a boundary layer).
We can apply a grid mapping function to map a uniform mesh into a non-uniform mesh that concentrates grid points toward the region of increased variation.
Note: We can think of the Chebyshev points as a uniform mesh mapped through the cosine function.

procedure:

1. build a uniform mesh t_i for $a \leq t \leq b$ for a chosen n,

$$t_i = a + ih, \quad i = 0 : n, \quad h = \frac{b-a}{n}$$

2. set the adapted mesh points x_i as the image of t_i under the mapping $g : x_i = g(t_i)$, where $g(t)$ is the grid mapping function

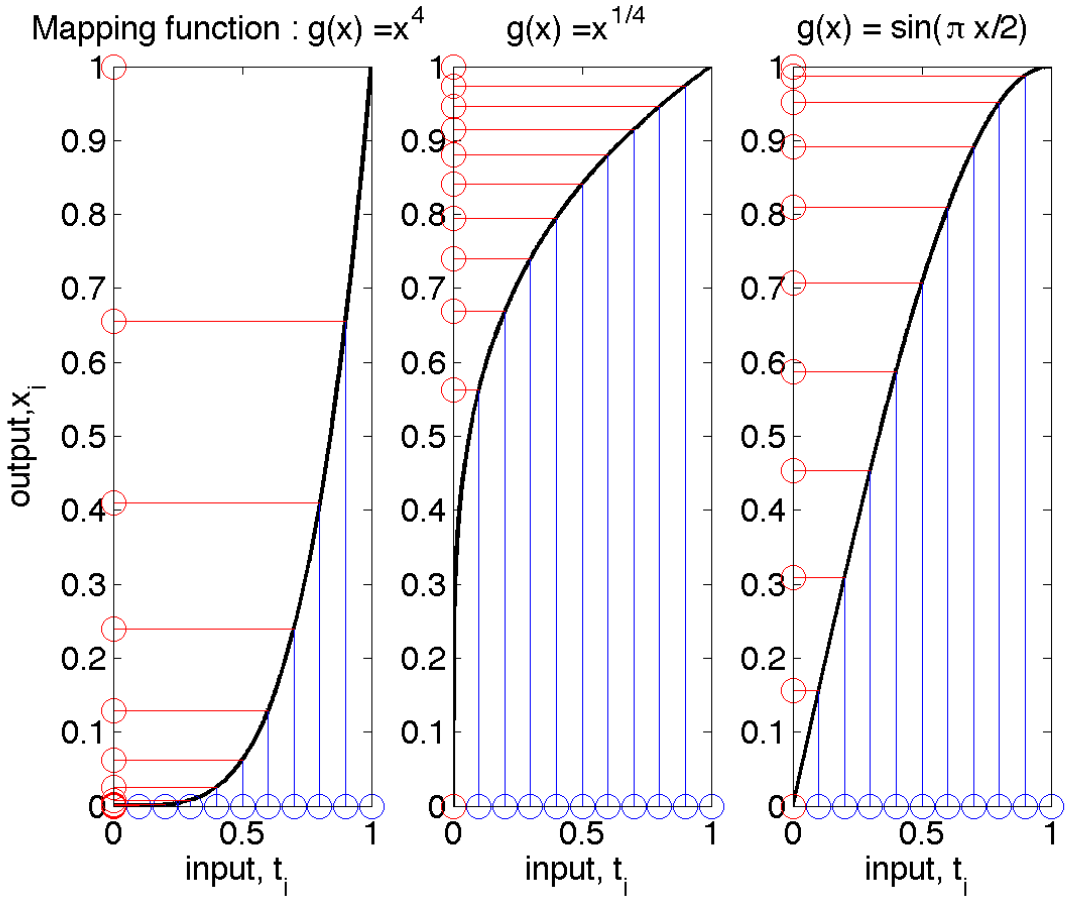
Grid mapping functions rules of thumb :

- the mapping function g should be monotonic (i.e., strictly increasing) and invertible on [a,b]

- for a given n, concentrate points where you need them most (where $|f'(x)|$ is large), decrease resolution where the integrand is not changing much (where $|f'(x)|$ is small).

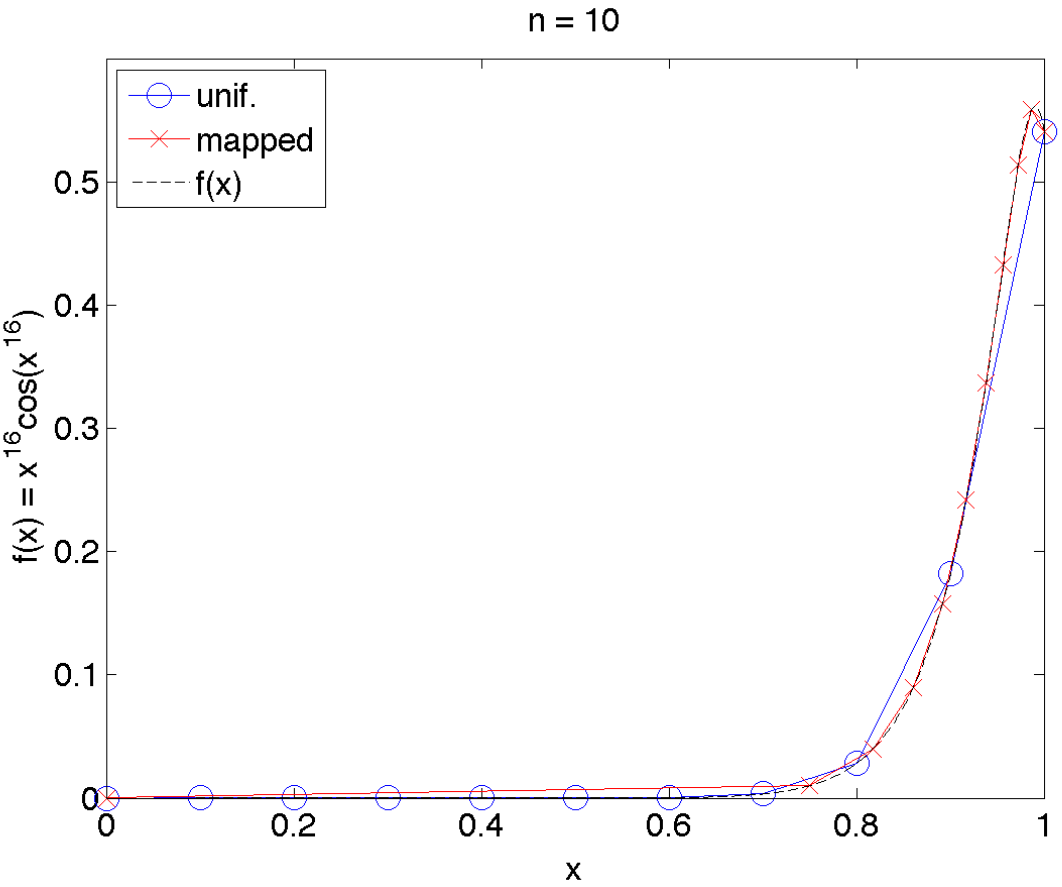
  The concavity or curvature of g controls the image points

- It helps to know the properties of basic functions!



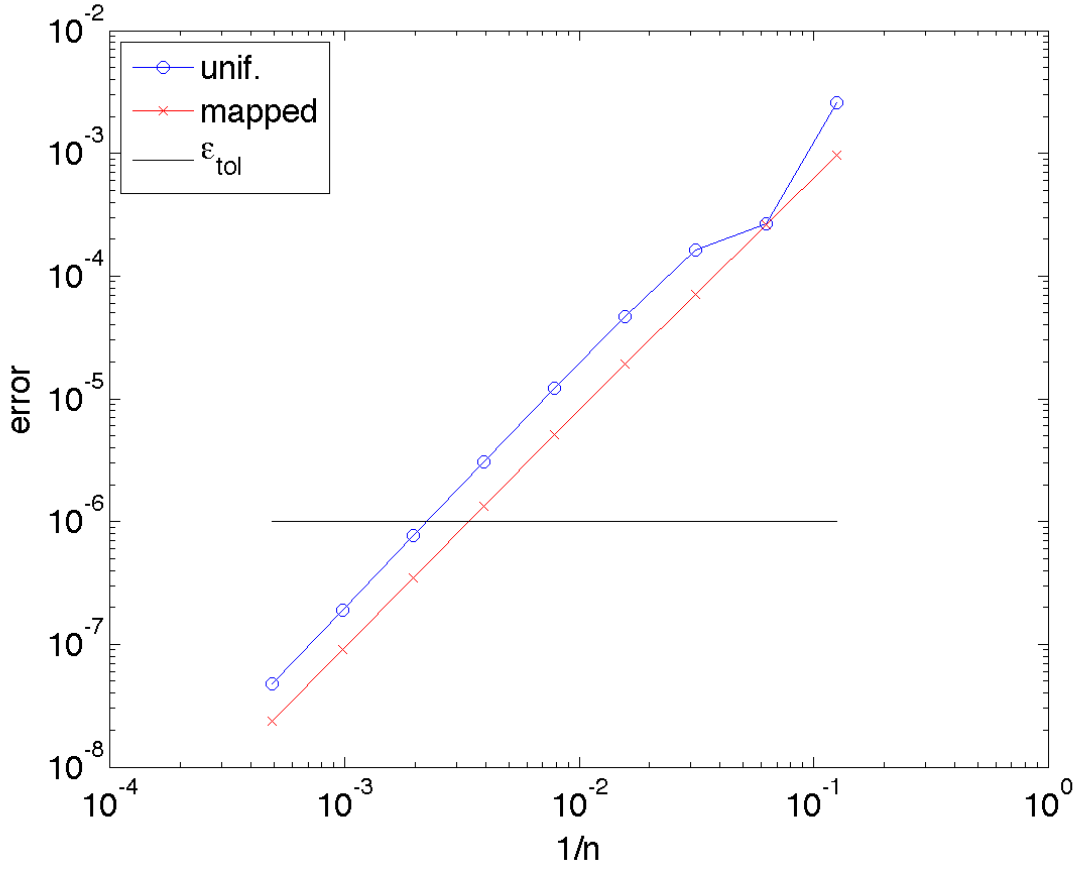
Example 4: Compute $\int_0^1 x^{16}\cos(x^{16})\,dx$ using the midpoint rule on (a) a uniform mesh, and (b) a mapped mesh using mapping function $g(x) = x^{1/8}$.

Note: the answer, to 20 digits of precision is 0.04912172951763908619 8.



Uniform Meshes			
n	$M(h)$	error	error/h^2
1	0.000015258789	0.049106470729	0.049106470729
2	0.005011046299	0.044110683219	0.176442732875
4	0.029446844621	0.019674884896	0.314798158341
8	0.046547569679	0.002574159839	0.164746229678
16	0.049390343819	0.000268614302	0.068765261203
32	0.049283507311	0.000161777794	0.165660460676
64	0.049168590195	0.000046860678	0.191941336088
128	0.049133849141	0.000012119623	0.198567910638
256	0.049124784728	0.000003055210	0.200226239298
512	0.049122494902	0.000000765384	0.200640894947
1024	0.049121920963	0.000000191445	0.200744562950
2048	0.049121777385	0.000000047867	0.200770480296
4096	0.049121741485	0.000000011967	0.200776959420
8192	0.049121732509	0.000000002992	0.200778577477

Mapped Meshes			
n	$M_g(h)$	error	error$/h^2$
1	0.000015258789	0.049106470729	0.049106470729
2	0.036818371688	0.012303357830	0.049213431318
4	0.045687188515	0.003434541003	0.054952656043
8	0.048162794242	0.000958935275	0.061371857618
16	0.048857968518	0.000263761000	0.067522815941
32	0.049050217883	0.000071511635	0.073227914059
64	0.049102570439	0.000019159079	0.078475585571
128	0.049116645794	0.000005083724	0.083291734766
256	0.049120391182	0.000001338336	0.087709166046
512	0.049121379480	0.000000350037	0.091760219122
1024	0.049121638465	0.000000091052	0.095475113820
2048	0.049121705942	0.000000023575	0.098881703045
4096	0.049121723438	0.000000006080	0.102005563909
8192	0.049121727955	0.000000001563	0.104870158248



△

2.3.2 adaptive strategy 2: adaptive mesh refinement (AMR)

Grid mapping functions work well when you know *in advance* where the regions of the problem domain are that will require a greater number of points to resolve.

Question 2: Can we develop algorithms that figure out where those regions are, without advance knowledge, and automatically put more points there?

idea: mesh refinement criteria

1. loop over each subinterval, if subinterval j exceeds a refinement tolerance, divide that interval
2. leave coarse subintervals that pass criteria alone

Question 3: How do we decide what useful refinement criteria are?

1. Local error estimates
   - create a local estimate of truncation error; for example, we know the local error of a piecewise polynomial interpolation on each of its subintervals; we can integrate that local polynomial to get a local estimate of error.
     If that estimate exceeds a tolerance, refine that subinterval.
   - For an example, **read pages 505-510**
2. physical criteria
   - perhaps you can choose regions that require high resolution by looking at the physical interpretations of the integral, or the integrand, or the data you already have.
   - for example, to get a good approximation of velocity, we would refine regions where acceleration has a large magnitude
3. geometric criteria (particularly in multiple space dimensions)
   - refine regions where the orientation of nearby mesh points is undesirable by some geometric measure, like curvature or extreme angles
   - leads to computational geometry algorithms

Example 5: Use AMR to compute $\int_0^1 x^{16}\cos(x^{16})\,dx$.

step 1: choose refinement criterion.

$\int_{x_j}^{x_{j+1}} f'(x)\,dx = f(x_{j+1}) - f(x_j) = \Delta f_j$ is the net change in $f(x)$ on the subinterval

$\frac{1}{x_{j+1}-x_j}\int_{x_j}^{x_{j+1}} f'(x)\,dx = \overline{f'}(x_j)$ is the average value of $f'(x)$ on the subinterval

possible refinement criteria:

1. $\left|\frac{f(x_{j+1})-f(x_j)}{x_{j+1}-x_j}\right| \approx \left|\overline{f'}(x_j)\right|$
2. $|f(x_{j+1})-f(x_j)| = \Delta f_j$
3. $|f(x_{j+1})-f(x_j)|\cdot(x_{j+1}-x_j) = \Delta f_j \cdot \Delta x_j$

Each of these have potential drawbacks:

1. may become difficult to evaluate, subject to cancellation error as $x_j \to x_{j+1}$ and overflow
2. is susceptible to grid aliasing; example: suppose $f(x) = \sin 20\pi x$, and your initial grid for $x \in [0, 1]$ is uniform with mesh spacing $h = \frac{1}{20}$; the criterion will never be triggered.
3. may make sense and work numerically, but what is the physical interpretation of that quantity?

step 2: Apply that criterion over a mesh, and refine until every subinterval passes, or until the number of mesh points reaches its maximum allowable amount (runs out of memory), or until the maximum number of refinement steps is reached

step 3: compute the integral using a simple method on the refined mesh (i.e., trapezoid, midpoint, Simpson's rule, cubic piecewise integration etc.)

△

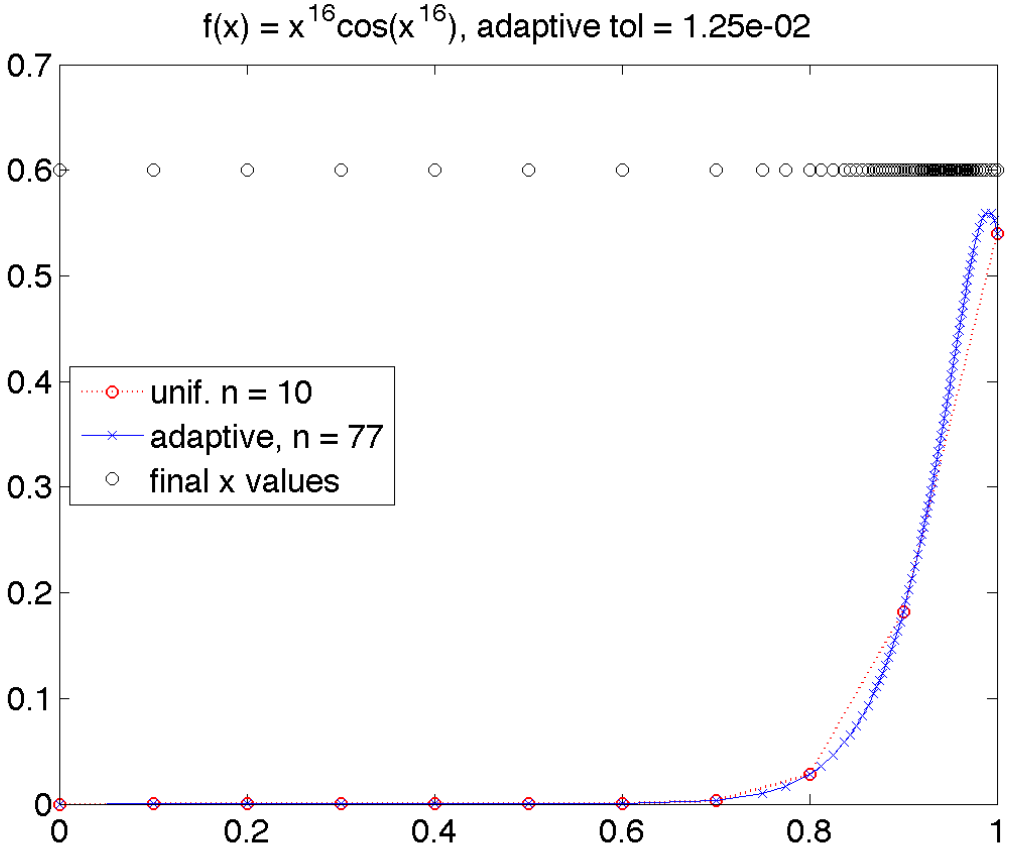
```
clear;
%% adaptive mesh refinement

diffTol = 0.01;
maxIter = 20;
nmax = 2000;
%
%    allocate memory
%
x = zeros(nmax,1);
newx = zeros(nmax,1);
f = zeros(nmax,1);
newf = zeros(nmax,1);
%
%    define initial mesh
%
n = 10;
for j=1:n+1
    x(j) = (j-1)/n;
    f(j) = x(j)^16*cos(x(j)^16);
end
x0 = x(1:n+1);
f0 = f(1:n+1);
n0 = n;
%
%    apply adaptive refinement
%
for k = 1:maxIter
    newn = 0;
    for j=1:n
```

```
        if ( abs(f(j+1) - f(j))  > diffTol )
            % divide this subinterval
            newx(newn+1) = x(j);
            newx(newn+2) = 0.5*(x(j) + x(j+1));
            % define new function values
            newf(newn+1) = f(j);
            newf(newn+2) = newx(newn+2)^16*cos(newx(newn+2)^16);
            % update insertion point
            newn = newn + 2;
        else
            newx(newn+1) = x(j);
            newf(newn+1) = f(j);
            newn = newn+1;
        end
    end
    newx(newn+1) = x(n+1);
    newf(newn+1) = f(n+1);
    n = newn;
    %
    % check convergence criterion
    %
    if ( sum(x(1:n+1) - newx(1:n+1)) == 0 )
        datastring = sprintf('mesh converged at step %d, n = %d',k,n);
        disp(datastring);
        x = newx;
        f = newf;
        break
    %
    % check memory limits
    %
    elseif (n >= nmax)
        datastring = sprintf('memory limit reached at step %d, n = %d',k,n);
        disp(datastring);
        x = newx;
        f = newf;
        break
    else
        x = newx;
        f = newf;
    end
end
%
%   check for non-convergence
%
if ( k == maxIter )
   datastring = sprintf('mesh convergence not achieved by step %d',maxIter);
   disp(datastring);
end
```

```
%% plot

figure(1);clf;
plot(x0,f0,'r:o','LineWidth',1.25);hold on;
xe = 0:0.001:1;
fe = xe.^16.*cos(xe.^16);
plot(x(1:n+1),f(1:n+1),'b-x');
plot(x(1:n+1),0.6*ones(n+1,1),'ko');
set(gca,'FontSize',16);
datastring1 = sprintf('unif. n = %d',n0);
datastring2 = sprintf('adaptive, n = %d',n);
titlestring = sprintf('adaptive tol = %4.2e',diffTol);
legend(datastring1,datastring2,'final x values','Location','West');
title(titlestring);
```



2.4 orthogonal polynomials

Text : section 5.4

recall: The inner product of two vectors in n-dimensional space, also known as the dot product

is

$$x \cdot y = x^T y = \sum_{i=1}^{n} x_i y_i$$

Definition 2. The inner product of two functions, $f(x)$ and $g(x)$, on the interval [-1,1] is

$$\langle f, g \rangle = \int_{-1}^{1} f(x) g(x)\, dx$$

properties of the inner product :

1. $\langle f, f \rangle \geq 0$ and $\langle f, f \rangle^{1/2} = \|f\|$: definition of the norm of the function f on [-1,1].
   $\|f\| = 0 \Leftrightarrow f = 0$
2. $\langle f, \alpha g + h \rangle = \alpha \langle f, g \rangle + \langle f, h \rangle$

Definition 3. Two functions, $f(x)$ and $g(x)$, are said to be orthogonal if $\langle f, g \rangle = 0$.

Example 6:

1. $\langle \sin \pi x, \cos \pi x \rangle = \displaystyle\int_{-1}^{1} \sin \pi x \cos \pi x\, dx = \frac{1}{2\pi} \sin^2 \pi x \Big|_{-1}^{1} = 0$: orthogonal
2. $\langle 1, x \rangle = \displaystyle\int_{-1}^{1} x\, dx = 0$: orthogonal
3. $\langle 1, x^2 \rangle = \displaystyle\int_{-1}^{1} x^2\, dx = \frac{2}{3}$: not orthogonal

△

Gram-Schmidt process : create orthogonal functions from a given set of basis functions.
For example, we can create orthogonal polynomials by applying the Gram-Schmidt process to the set $\{1, x, x^2, x^3, x^4, \dots\}$.
This will lead to an orthogonal set of polynomials $\{P_0(x), P_1(x), P_2(x), \dots\}$ called the Legendre polynomials.
Note : The roots of the Legendre polynomials provide the optimal points for minimizing the 2-norm of interpolation error.
Recall that the Chebyshev points provide the optimal interpolation points for minimizing the ∞-norm of interpolation error; the Chebyshev points are the roots of the orthogonal set of Chebyshev polynomials.
The Gram-Schmidt process for functions works just like the Gram-Schmidt process for creating orthogonal sets of vectors. At each step, we subtract the components of the next basis element that lie in the same "direction" as the basis elements that precede it, where direction is determined by the inner product.

$P_0(x) = 1$: note $\|P_0(x)\| = \langle 1, 1 \rangle^{1/2} = \sqrt{2}$

$$P_1(x) = x - \frac{\langle x, P_0 \rangle}{\|P_0\|^2} P_0(x) = x, \qquad \|P_1\| = \int_{-1}^{1} x^2\, dx = \frac{2}{3}$$

$$P_2(x) = x^2 - \frac{\langle x^2, P_1 \rangle}{\|P_1\|^2} P_1 - \frac{\langle x^2, P_0 \rangle}{\|P_0\|^2} P_0 = x^2 - \frac{1}{3}$$

$$\text{check}: \langle P_2, P_0\rangle = \int_{-1}^{1} x^2 - \frac{1}{3}\,dx = 0, \qquad \langle P_2, P_1\rangle = \int_{-1}^{1} x\left(x^2 - \frac{1}{3}\right)\,dx = 0$$

$$P_3(x) = x^3 - \frac{\langle x^3, P_2\rangle}{\|P_2\|^2}P_2 - \frac{\langle x^3, P_1\rangle}{\|P_1\|^2}P_1 - \frac{\langle x^3, P_0\rangle}{\|P_0\|^2}P_0 = x^3 - \frac{3}{5}x$$

$$\langle x^3, P_0\rangle = \int_{-1}^{1} x^3\,dx = 0$$

$$\langle x^3, P_1\rangle = \int_{-1}^{1} x^3 \cdot x\,dx = \frac{2}{5}$$

$$\langle x^3, P_2\rangle = \int_{-1}^{1} x^3\left(x^2 - \frac{1}{3}\right)\,dx = 0$$

$$P_4(x) = x^4 - \frac{\langle x^4, P_3\rangle}{\|P_3\|^2}P_3 - \frac{\langle x^4, P_2\rangle}{\|P_2\|^2}P_2 - \frac{\langle x^4, P_1\rangle}{\|P_1\|^2}P_1 - \frac{\langle x^4, P_0\rangle}{\|P_0\|^2}P_0 = \cdots$$

$$\vdots$$

Notes:

- The Gram-Schmidt process creates a set of orthogonal polynomials that are unique up to a scaling factor; just like we would normalize orthogonal vectors in a vector space, we apply some standardization to the above polynomials...
- To create "unit vectors" in this function space, we would divide each polynomial by its norm.
- Instead, we apply a more common standardization and scale each polynomial so that it is a monic polynomial. A polynomial $p(x)$ is said to be monic if it satisfies $p(1) = 1$.

  Thus, we multiply each of the above polynomials by the appropriate scaling factor to get $P_i(1) = 1$. For example, $P_2(x) = x^2 - \frac{1}{3} \Rightarrow P_2(1) = \frac{2}{3}$, so we would multiply the $P_2(x)$ above by $\frac{3}{2}$ to get the standard form : $P_2(x) = \frac{3}{2}(x^2 - \frac{1}{3})$.
- These polynomials can also be found using Rodrgues formula :

  $$P_k(x) = \frac{1}{2^k k!}\frac{d^k}{dx^k}\left((x^2-1)^k\right)$$

  or the recursion relation :

  $$(k+1)P_k(x) = (2k+1)xP_k(x) - kP_{k-1}(x)$$
- Orthogonal polynomials in general are a very important concept in mathematics. They are intimately related to the field of spectral methods. We have spoken already about the Legendre and Chebyshev sets of orthogonal polynomials; other examples of orthogonal sets of polynomials include the Hermite polynomials, Laguerre polynomials, Jacobi polynomials, etc...

Tuesday, 11/26/13

Legendre polynomials

1. $P_n(x)$ is a polynomial of degree n that can be found by applying the Gram-Schmidt process to the standard basis $\{1, x, x^2, x^3, \dots\}$.

2. Typically the Legendre polynomials are presented as monic polynomials, that is the polynomials resulting from Gram-Schmidt are multiplied by a scalar so that each of them satisifies $P_n(1) = 1$.

3. Any polynomial $q(x)$ of degree $\leq n$ can be written as $q(x) = \sum_{k=1}^{n} c_k P_k(x)$, i.e., the Legendre polynomials $\{P_0(x), P_1(x), \dots, P_n(x)\}$ form a basis for the vector space of polynomials of degree $\leq n$.

4. The first few Legendre polynomials are

n	$P_n(x)$
0	1
1	x
2	$\frac{1}{2}(3x^2 - 1)$
3	$\frac{1}{2}(5x^3 - 3x)$
4	$\frac{1}{8}(35x^4 - 30x^2 + 3)$
5	$\frac{1}{8}(63x^5 - 70x^3 + 15x)$
6	$\frac{1}{16}(231x^6 - 315x^4 + 105x^2 - 5)$
$\vdots$	$\vdots$

Back to numerical integration...

2.5 Gaussian quadrature

text : section 6.6

So far we have developed quadrature rules

$$\int_a^b f(x)\,dx \approx \sum_{i=0}^{n} w_i f(x_i)$$

for a known set of mesh points x_i (also called abscissas) and a given set of quadrature weights w_i.

Example : Trapezoid rule, uniform mesh

1. The abscissas are defined as $xi = a + ih, \quad h = \frac{1}{n}, \quad i = 0 : n.$

2. The weights are $w_i = 1, \quad i = 2 : n-1, \quad w_0 = w_n = \frac{1}{2}$

Question 4: Can we design a quadrature scheme by choosing both x_i and w_i to achieve the best possible accuracy for a given n?
In other words, given a positive integer n can we choose $x_i \in [-1, 1]$ and w_i such that

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

for all $f(x) \in \{1, x, x^2, \ldots, x^{2n-1}\}$?

△

Notes:

- Be careful of the indexing! In this section (unlike the previous sections), the first index is 1, not zero, so the first point is x_1, not x_0, meaning we will use a total of n points, not $n+1$.
- We have $2n$ variables : $x_1, x_2, \ldots, x_n$ and $w_1, w_2, \ldots, w_n$.

Example 7: 1 point Gaussian quadrature, generic interval

Derive a quadrature rule $\sum_{i=1}^{n} w_i f(x_i)$ for $n = 1$ that is exact for $f(x) = 1, x$ on the interval $x \in [a, b]$. In other words, this quadrature rule should be exact for all linear functions.

We can set up a system of 2 (nonlinear) equations in the unknowns x_1 and w_1.

$$\int_a^b 1\,dx = w_1 = b - a$$
$$\int_a^b x\,dx = w_1 x_1 = \frac{1}{2}(b^2 - a^2)$$
$$\Rightarrow w_1 = b - a, \quad x_1 = \frac{b-a}{2}$$

The resulting quadrature formula is the midpoint rule

$$\int_a^b f(x)\,dx \approx (b-a) \cdot f\left(\frac{1}{2}(b-a)\right)$$

which has truncation error equal to $\frac{(b-a)^2}{24} f''(\xi)$ for some $\xi \in [a, b]$.

△

Note: pth order accuracy $\Leftrightarrow$ exact quadrature for polynomials of degree $\leq p - 1$

Example 8: 2 point Gaussian quadrature, standard interval

Derive a quadrature rule $\sum_{i=1}^{n} w_i f(x_i)$ using $n = 2$ that is exact for polynomials of degree 3 or less.

Just like the previous example, the quadrature rule should be exact for all $f(x) \in \{1, x, x^2, x^3\}$; we can build the system of 4 equations : $\int_{-1}^{1} x^k \, dx = \sum_{i=1}^{n} w_i x_i^k$ for $k = 0, 1, 2, 3$.

$$\int_{-1}^{1} 1 \, dx = w_1 + w_2 = 2$$
$$\int_{-1}^{1} x \, dx = w_1 x_1 + w_2 x_2 = 0$$
$$\int_{-1}^{1} x^2 \, dx = w_1 x_1^2 + w_2 x_2^2 = \frac{2}{3}$$
$$\int_{-1}^{1} x^3 \, dx = w_1 x_1^3 + w_2 x_2^3 = 0$$

Note that one advantage of choosing this interval is that it is symmetric about the origin; it would make sense if our grid points and weights were also symmetric $\Rightarrow x_2 = -x, \;\; w_1 = w_2$. Plugging $x_1 = -x_2$ and $w_1 = w_2$ into the above equations, we find that $w_1 = w_2 = 1$ from equation 1. The second and fourth equations are satisfied exactly, and the third equation gives $x_1 = -\frac{1}{\sqrt{3}}, \; x_2 = \frac{1}{\sqrt{3}}$.

The 2-point Gaussian quadrature formula for the interval $-1 \leq x \leq 1$ is therefore

$$\int_{-1}^{1} f(x) \, dx \approx f\left(-\frac{1}{\sqrt{3}}\right) + f\left(\frac{1}{\sqrt{3}}\right)$$

△

Theorem 4 (Gaussian quadrature)**.**

1. *The Legendre polynomial $P_n(x)$ has n distinct roots in $(-1, 1)$; call them $x_1, x_2, \ldots, x_n$. These will be the integration points, aka the abscissas, for Gaussian quadrature.*

2. *There exists a set of constants c_i, $i = 1 : n$, where $n \geq 1$, such that the integration rule*

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

is exact for all polynomials of degree $\leq 2n - 1$.

3. *Error in the Gaussian quadrature formula is proportional to the $2n$ derivative of $f(x)$, that is*

$$\int_{-1}^{1} f(x) \, dx = \sum_{i=1}^{n} c_i f(x_i) + C_n f^{(2n)}(\xi)$$

where ξ is some point in $[-1, 1]$ and C_n is a constant.

Proof. 1. We assume $n \geq 1$; since the Legendre polynomials are orthogonal,

$\langle P_n, P_0 \rangle = \int_{-1}^{1} P_n(x)\,dx = 0 \quad \Rightarrow \quad P_n(x)$ changes sign at least once in $(-1,1)$. Let $x_1, \ldots, x_j$ be the points in $(-1,1)$ where $P_n(x)$ changes sign, so $1 \leq j \leq n$.

Define $q(x) = (x-x_1)(x-x_2)\cdots(x-x_j)$ and note that q also changes sign at $x_1, x_2, \ldots, x_j$, and that $\deg q = j$.

Now consider the intervals $(-1, x_1), (x_1, x_2), \ldots, (x_j, 1)$, and note that $P_n(x)$ and q must either always have the same sign or always have the opposite on each individual interval. In either case, $\langle P_n, q \rangle = \int_{-1}^{1} P_n(x)q(x)\,dx \neq 0$.

This implies that $\deg q \geq n$, because $P_n(x)$ is orthogonal to polynomials of degree $< n$ by construction. So if $\deg q < n$, then $\langle P_n, q \rangle = 0$; since $\langle P_n, q \rangle \neq 0$, we must have $\deg q \geq n$.

$\deg q = j \leq n$ and $\deg q = j \geq n \Rightarrow \deg q = n$, and therefore that $P_n(x)$ has n distinct zeros in (-1,1).

2. Let $f(x)$ be a polynomial of degree $\leq 2n-1$.

   <u>case 1:</u> $\deg f \leq n-1$

   $f(x) = \sum_{i=1}^{n} f(x_i)L_i(x)$: Lagrange interpolating polynomial at $x_1, x_2, \ldots, x_n$.

   $$\int_{-1}^{1} f(x)\,dx = \int_{-1}^{1} \left(\sum_{i=1}^{n} f(x_i)L_i(x) \right) dx = \sum_{i=1}^{n} f(x_i) \left(\int_{-1}^{1} L_i(x)\,dx \right) = \sum_{i=1}^{n} c_i f(x_i)$$

   where $c_i = \int_{-1}^{1} L_i(x)\,dx$.

   <u>case 2:</u> $\deg f \leq 2n-1$

   write $f(x)$ as $f(x) = q(x)P_n(x) + r(x)$, where $q(x)$ is the quotient and $r(x)$ is the remainder after the division of $f(x)$ by the nth Legendre polynomial P_n.

   $\Rightarrow \deg q \leq n-1, \qquad \deg r \leq n-1$

   $\Rightarrow f(x_i) = q(x_i)P_n(x_i) + r(x_i) = r(x_i)$ because the x_i are roots of P_n

   $$\int_{-1}^{1} f(x)\,dx = \int_{-1}^{1} q(x)P_n(x)\,dx + \int_{-1}^{1} r(x)\,dx = \langle q, P_n \rangle + \int_{-1}^{1} r(x)\,dx$$

   $\langle q, P_n \rangle = 0$ because $\deg q < n \Rightarrow q$ is orthogonal to P_n.

   Since $\deg r \leq n-1$, case 1 applies and $\int_{-1}^{1} r(x)\,dx = \sum_{i=1}^{n} c_i f(x_i)$

   $$\Rightarrow \int_{-1}^{1} f(x)\,dx = \sum_{i=1}^{n} c_i f(x_i)$$

3. We omit the proof of 3 for the sake of time. It can be found by consider the connection between Gaussian quadrature and the Hermite interpolating polynomial. Also, exercises 31 and 32 from the text provide a walk-through.

□

Notes:

- The above defines the roots of the Legendre polynomials $P_n(x)$ as the mesh points, quadrature points, or abscissas (abscissae?) for Gaussian quadrature. The quadrature weights are defined as

$$w_i = \int_{-1}^{1} L_i(x)\,dx$$

where $L_i(x)$ is the Lagrange polynomial associated with the point x_i.

- The reason we use a standardized interval is so that we don't have to calculate x_i and c_i every time we want to use Gaussian quadrature : we can store them in a table and just look them up when we need them.

n	x_i	w_i
1	0	2
2	$\pm\sqrt{\frac{1}{3}}$	1
3	$-\sqrt{\frac{3}{5}},\ 0,\ \sqrt{\frac{3}{5}}$	$\frac{5}{9},\ \frac{8}{9},\ \frac{5}{9}$
4	$-\sqrt{\frac{3+2\sqrt{6/5}}{7}},\ -\sqrt{\frac{3-2\sqrt{6/5}}{7}},\ \sqrt{\frac{3-2\sqrt{6/5}}{7}},\ \sqrt{\frac{3+2\sqrt{6/5}}{7}}$	$-\frac{18-\sqrt{30}}{36},\ \frac{18+\sqrt{30}}{36},\ \frac{18+\sqrt{30}}{36},\ \frac{18-\sqrt{30}}{36}$
$\vdots$	$\vdots$	$\vdots$

- The roots of $P_n(x)$ do not include the endpoints $x = \pm 1$; hopefully this helps explain the different indexing in this section: In all other sections we placed grid points at the boundaries of the domain (more later).

Example 9: Calculate $\int_0^1 e^{-x^2}\,dx$ using Gaussian quadrature with $n = 1, 2, 3$.

1. Change variables so that $a = -1,\ b = 1$ (to use the standard interval of integration)

   $t = 2x + 1 \quad \Rightarrow \quad x = \frac{1}{2}(t+1),\ dx = \frac{1}{2}dt$

   $$\int_0^1 e^{-x^2}\,dx = \frac{1}{2}\int_{-1}^{1} e^{-\frac{1}{4}(t+1)^2}\,dt$$

2. Compute using the standard interval Gaussian quadrature formulas.

   $$n = 1 : \frac{1}{2}\int_{-1}^{1} e^{-\frac{1}{4}(t+1)^2}\,dt \approx \frac{1}{2}\left(2\cdot e^{-\frac{1}{4}}\right) = 0.\underline{7}78800783071405$$

   $$n = 2 : \frac{1}{2}\int_{-1}^{1} e^{-\frac{1}{4}(t+1)^2}\,dt \approx \frac{1}{2}\left(e^{-\frac{1}{4}(1-\frac{1}{\sqrt{3}})^2} + e^{-\frac{1}{4}(1+\frac{1}{\sqrt{3}})^2}\right) = 0.\underline{746}594688282860$$

   $$n = 3 : \frac{1}{2}\int_{-1}^{1} e^{-\frac{1}{4}(t+1)^2}\,dt \approx \frac{1}{2}\left(\frac{5}{9}e^{-\frac{1}{4}(1+\sqrt{\frac{3}{5}})^2} + \frac{8}{9}e^{-\frac{1}{4}} + \frac{5}{9}e^{-\frac{1}{4}(1-\sqrt{\frac{3}{5}})^2}\right) = 0.\underline{7468}14584191256$$

Gaussian quadrature with 3 points approximately as accurate than the trapezoid rule with 9 points, or Romberg's method $R_2(0.25)$.

△

Notes:

- For algorithms that rely on integration (such as integral transform methods) it is common to define the mesh points as the Gaussian quadrature points for a given n.
- In the age of parallel computing, however, it is sometimes inefficient to calculate global integrals in multiple space dimensions; instead, many advanced algorithms use finite element methods.

  These methods divide the domain into several subdomains – the elements – and choose an order of accuracy to apply on each element, which then defines a local mesh within an element. The most recent climate model published by the National Center for Atmospheric Research in Boulder, CO., discretizes the sphere as a collection of thousands of quadrilaterals. Within each of these quadrilaterals, a Gaussian quadrature mesh is defined that typically integrates 6th order polynomials exactly.

  It is known as the Community Atmosphere Model - Spectral Element (CAM-SE) model; a version of it runs on Flux here at Michigan.
- We began this section by setting up a system of $2n$ equations for the unknowns x_i and w_i, and chose the grid points x_i to be optimal in the sense that the quadrature scheme integrates polynomials of degree $2n-1$ exactly. This procedure is another example of the method of undetermined coefficients. If some, but not all, grid points are fixed, this method can still be used to find the optimal locations for the grid points that are yet to be determined. They may no longer correspond to the roots of an orthogonal set of polynomials, which may take them longer to find, but they can of course still be found.

Question 5: So far we have assumed that the integrands $f(x)$ are well-approximated by polynomials, and that the intervals of integration are finite. How do these quadrature rules perform when approximating improper integrals – integrals with singularities within the interval or infinite intervals of integration?

For example:

1. $\displaystyle\int_0^1 \frac{1}{\sqrt{x}}\,dx$
2. $\displaystyle\int_0^\infty f(x)e^{-x}\,dx$

2.6 Gauss-Laguerre quadrature

$$\int_0^\infty f(x)e^{-x}\,dx$$

strategies:

1. recall that $\displaystyle\int_0^\infty f(x)e^{-x}\,dx = \lim_{M\to\infty}\int_0^M f(x)e^{-x}\,dx$. Choose a finite M and approximate the integral on a truncated domain.
2. Apply a mapping function to map $[0,\infty) \to [0,1)$
3. define a new inner product : $\langle f, g\rangle = \int_0^\infty f(x)g(x)e^{-x}\,dx$

The Laguerre polynomials are orthogonal with respect to this inner product, and they are obtained by applying Gram-Schmidt to $\{1, x, x^2, \dots\}$ using this inner product.

$$\begin{aligned}
\mathcal{L}_0(x) &= 1\\
\mathcal{L}_1(x) &= x - 1\\
\mathcal{L}_2(x) &= x^2 - 4x + 2\\
&\vdots
\end{aligned}$$

Let $x_1, \dots, x_n$ be the roots of $\mathcal{L}_n(x)$ and set $c_i = \int_0^\infty \mathcal{L}_i(x)e^{-x}\,dx$

then $\displaystyle\int_0^\infty f(x)e^{-x}\,dx \approx \sum_{i=1}^n c_i f(x_i)$ is exact for polynomials of degree $\le 2n-1$.

Example 10: 2-point Gauss-Laguerre rule.

$$\int_0^\infty f(x)e^{-x}\,dx \approx \left(\frac{\sqrt{2}+1}{2\sqrt{2}}\right) f(2-\sqrt{2}) + \left(\frac{\sqrt{2}-1}{2\sqrt{2}}\right) f(2+\sqrt{2})$$

△