Lab # 4 Solutions

Sam Fleischer

Thurs. Apr. 14, 2015

Contents

Problem 1

Write a function called `trapzoid(f, a, b, n)` *that implements the composite trapezoidal rule. Your function should take as input the name of the function to integrate,* `f`*, the end-points of the interval of integration,* `a` *and* `b`*, and the number of points* `n` *of subintervals to divide the integral of integration into. Use the function to approximate the following integrals with values of* `n` $= 10, 20, 50, 100,$ *and* 200*. In each case indicate the error. Also, make sure your implementation of the trapezoidal rule does not evaluate the function being integrated more than once at each* x *value.*

(a) $\displaystyle\int_0^{\pi} \sin x \, dx$

The exact value is

$$(-\cos x)\Big|_0^{\pi} = (-\cos(\pi)) - (-\cos(0)) = 1 - (-1) = 2$$

The error is bounded by $|E_T| \leq \dfrac{K(b-a)^3}{12n^2}$, where $a = 0$, $b = \pi$, and $K = \left|\max\limits_{x\in(0,\pi)} \dfrac{d^2}{dx^2} \sin x\right| = 1$. So, $E_T \leq \dfrac{\pi^3}{12n^2}$

n	T_n	**Error Bound**	**Actual Error**
10	1.9835235375	$\frac{\pi^3}{12(10)^2} = 0.025838564$	0.01647646250000001
20	1.9958859727	$\frac{\pi^3}{12(20)^2} = 0.006459641$	0.004114027299999989
50	1.9993419831	$\frac{\pi^3}{12(50)^2} = 0.001033543$	0.0006580168999998914
100	1.9998355039	$\frac{\pi^3}{12(100)^2} = 0.000258386$	0.0001644961000002224
200	1.9999588765	$\frac{\pi^3}{12(200)^2} = 6.459640975 \times 10^{-5}$	$4.112349999996212 \times 10^{-05}$

(b) $\displaystyle\int_0^{\pi} \cos x \, dx$

The exact value is

$$(\sin x)\Big|_0^{\pi} = (\sin(\pi)) - (\sin(0)) = 0 - 0 = 0$$

The error is bounded by $|E_T| \leq \dfrac{K(b-a)^3}{12n^2}$, where $a = 0$, $b = \pi$, and $K = \left|\max\limits_{x\in(0,\pi)} \dfrac{d^2}{dx^2} \cos x\right| = 1$. So, $E_T \leq \dfrac{\pi^3}{12n^2}$

n	T_n	**Error Bound**	**Actual Error**
10	3.885781×10^{-16}	$\frac{\pi^3}{12(10)^2} = 0.025838564$	3.885781×10^{-16}
20	1.665335×10^{-16}	$\frac{\pi^3}{12(20)^2} = 0.006459641$	1.665335×10^{-16}
50	$-2.775558 \times 10^{-17}$	$\frac{\pi^3}{12(50)^2} = 0.001033543$	2.775558×10^{-17}
100	$-4.093947 \times 10^{-16}$	$\frac{\pi^3}{12(100)^2} = 0.000258386$	4.093947×10^{-16}
200	6.591949×10^{-17}	$\frac{\pi^3}{12(200)^2} = 6.459640975 \times 10^{-5}$	6.591949×10^{-17}

(c) $\displaystyle\int_0^{\pi/2} \sin x \, dx$

The exact value is

$$(-\cos x)\Big|_0^{\frac{\pi}{2}} = (-\cos(\tfrac{\pi}{2})) - (-\cos(0)) = 0 - (-1) = 1$$

The error is bounded by $|E_T| \leq \dfrac{K(b-a)^3}{12n^2}$, where $a = 0$, $b = \frac{\pi}{2}$, and $K = \left| \max\limits_{x \in (0, \frac{\pi}{2})} \dfrac{d^2}{dx^2} \sin x \right| = 1$. So, $E_T \leq \dfrac{\pi^3}{96n^2}$

n	T_n	**Error Bound**	**Actual Error**
10	0.9979430	$\frac{\pi^3}{96(10)^2} = 0.00322982$	0.002057
20	0.9994859	$\frac{\pi^3}{96(20)^2} = 0.000807455$	0.0005141
50	0.9999178	$\frac{\pi^3}{96(50)^2} = 0.000129193$	8.220000×10^{-5}
100	0.9999794	$\frac{\pi^3}{96(100)^2} = 3.229820488 \times 10^{-5}$	2.060000×10^{-5}
200	0.9999949	$\frac{\pi^3}{96(200)^2} = 8.074551219 \times 10^{-6}$	5.100000×10^{-6}

(d) $\displaystyle\int_0^{\ln 3} e^x \, dx$

The exact value is

$$(e^x)\Big|_0^{\ln 3} = (e^{\ln 3}) - (e^0) = 3 - 1 = 2$$

The error is bounded by $|E_T| \leq \dfrac{K(b-a)^3}{12n^2}$, where $a = 0$, $b = \ln 3$, and

$K = \left| \max_{x \in (0, \ln 3)} \dfrac{d^2}{dx^2} e^x \right| = 3$. So, $E_T \leq \dfrac{(\ln 3)^3}{4n^2}$

n	T_n	**Error Bound**	**Actual Error**
10	2.0020111771	$\frac{(\ln 3)^3}{4(10)^2} = 0.003314922$	0.0020111771
20	2.0005028701	$\frac{(\ln 3)^3}{4(20)^2} = 0.000828731$	0.0005028701
50	2.0000804626	$\frac{(\ln 3)^3}{4(50)^2} = 0.000132597$	8.04626×10^{-5}
100	2.0000201158	$\frac{(\ln 3)^3}{4(100)^2} = 3.3149224 \times 10^{-5}$	2.01158×10^{-5}
200	2.0000050290	$\frac{(\ln 3)^3}{4(200)^2} = 8.287306001 \times 10^{-6}$	5.0290×10^{-6}

The following Python code was used to generate the above approximations:

```python
from __future__ import division
from math import sin, cos, pi, exp, log

def irange(start, stop, step):
    r = start
    while r <= stop:
        yield r
        r += step

def trap_area(b_1, b_2, h):
    return (1/2)*(b_1 + b_2)*h

def comp_trap_rule(f, a, b, n):
    h = (b-a)/n

    y_values = [f(a + i*h) for i in irange(0, n, 1)]

    approx = 0
    for i in xrange(0, len(y_values) - 1):
        approx += trap_area(y_values[i], y_values[i+1], h)
    return approx

def problem_1():
    for n in [10, 20, 50, 100, 200]:
        print "\n    n = %d" % n
        a = comp_trap_rule(sin, 0, pi, n)
        b = comp_trap_rule(cos, 0, pi, n)
        c = comp_trap_rule(sin, 0, (pi/2), n)
        d = comp_trap_rule(exp, 0, log(3), n)
```

```python
        print "         a = %.10e" % a
        print "         b = %.10e" % b
        print "         c = %.10e" % c
        print "         d = %.10e" % d

problem_1()
```

Listing 1: Problem 1 source code

The following is the output of the above Python code:

```
    n = 10
        a = 1.9835235375e+00
        b = 3.8857805862e-16
        c = 9.9794298635e-01
        d = 2.0020111771e+00

    n = 20
        a = 1.9958859727e+00
        b = 1.6653345369e-16
        c = 9.9948590525e-01
        d = 2.0005028701e+00

    n = 50
        a = 1.9993419831e+00
        b = -2.7755575616e-17
        c = 9.9991775194e-01
        d = 2.0000804626e+00

    n = 100
        a = 1.9998355039e+00
        b = -4.0939474033e-16
        c = 9.9997943824e-01
        d = 2.0000201158e+00

    n = 200
        a = 1.9999588765e+00
        b = 6.5919492087e-17
        c = 9.9999485958e-01
        d = 2.0000050290e+00
```

Problem 2

Wtie a function `midpoint(f, a, b, n)` *that implements the composite midpoint rule. Your function should take as input the name of the function to integrate,* `f`*, the endpoints of the interval of integration,* `a` *and* `b`*, and the number of points* `n` *of subintervals to divide the integral of integration into. Use the function to approximate the same integrals as in* **Problem 1** *with values of* `n` $= 10$, 20, *amd* 50. *In each case indicate the error.*

(a) $\displaystyle\int_0^{\pi} \sin x \; dx$

The error is bounded by $|E_M| \leq \dfrac{K(b-a)^3}{24n^2}$, where $a = 0$, $b = \pi$, and $K = \left|\max\limits_{x\in(0,\pi)} \dfrac{d^2}{dx^2} \sin x\right| = 1$. So, $E_T \leq \dfrac{\pi^3}{24n^2}$

n	M_n	**Error Bound**	**Actual Error**
10	2.0082484079	$\frac{\pi^3}{24(10)^2} = 0.012919282$	0.0082484079
20	2.0020576483	$\frac{\pi^3}{24(20)^2} = 0.00322982$	0.0020576483
50	2.0003290247	$\frac{\pi^3}{24(50)^2} = 0.000516771$	0.0003290247

(b) $\displaystyle\int_0^{\pi} \cos x \; dx$

The error is bounded by $|E_M| \leq \dfrac{K(b-a)^3}{24n^2}$, where $a = 0$, $b = \pi$, and $K = \left|\max\limits_{x\in(0,\pi)} \dfrac{d^2}{dx^2} \cos x\right| = 1$. So, $E_T \leq \dfrac{\pi^3}{24n^2}$

n	M_n	**Error Bound**	**Actual Error**
10	$-5.5511151231 \times 10^{-17}$	$\frac{\pi^3}{24(10)^2} = 0.012919282$	$5.5511151231 \times 10^{-17}$
20	$-1.3877787808 \times 10^{-16}$	$\frac{\pi^3}{24(20)^2} = 0.00322982$	$1.3877787808 \times 10^{-16}$
50	$3.0531133177 \times 10^{-16}$	$\frac{\pi^3}{24(50)^2} = 0.000516771$	$3.0531133177 \times 10^{-16}$

(c) $\displaystyle\int_0^{\pi/2} \sin x \; dx$

The error is bounded by $|E_M| \leq \dfrac{K(b-a)^3}{24n^2}$, where $a = 0$, $b = \frac{\pi}{2}$, and $K = \left| \max_{x\in(0,\frac{\pi}{2})} \dfrac{d^2}{dx^2} \sin x \right| = 1$. So, $E_T \leq \dfrac{\pi^3}{192n^2}$

n	M_n	**Error Bound**	**Actual Error**
10	1.0010288241	$\frac{\pi^3}{192(10)^2} = 0.00161491$	0.0010288241
20	1.0002570672	$\frac{\pi^3}{192(20)^2} = 0.000403728$	0.0002570672
50	1.0000411245	$\frac{\pi^3}{192(50)^2} = 6.4596 \times 10^{-5}$	4.11245×10^{-5}

(d) $\displaystyle\int_0^{\ln 3} e^x \, dx$

The error is bounded by $|E_M| \leq \dfrac{K(b-a)^3}{24n^2}$, where $a = 0$, $b = \ln 3$, and

$K = \left| \max_{x\in(0,\ln 3)} \dfrac{d^2}{dx^2} e^x \right| = 3$. So, $E_T \leq \dfrac{(\ln 3)^3}{8n^2}$

n	M_n	**Error Bound**	**Actual Error**
10	1.9989945632	$\frac{(\ln 3)^3}{8(10)^2} = 0.001657461$	0.0010054368
20	1.9997485744	$\frac{(\ln 3)^3}{8(20)^2} = 0.000414365$	0.0002514256
50	1.9999597689	$\frac{(\ln 3)^3}{8(50)^2} = 6.6298 \times 10^{-5}$	4.02311×10^{-5}

The following Python code was used to generate the above approximations:

```python
from __future__ import division
from math import sin, cos, pi, exp, log

def rect_area(b, h):
    return b*h

def comp_mdpt_rule(f, a, b, n):
    h = (b-a)/n

    approx = 0
    for i in xrange(0, n):
        approx += rect_area(h, f(a + i*h + (1/2)*h))
    return approx
```

```python

def problem_2():
    for n in [10, 20, 50]:
        print "\n    n = %d" % n
        a = comp_mdpt_rule(sin, 0, pi, n)
        b = comp_mdpt_rule(cos, 0, pi, n)
        c = comp_mdpt_rule(sin, 0, (pi/2), n)
        d = comp_mdpt_rule(exp, 0, log(3), n)
        print "        a = %.10e" % a
        print "        b = %.10e" % b
        print "        c = %.10e" % c
        print "        d = %.10e" % d


problem_2()
```

Listing 2: Problem 2 source code

The following is the output of the above Python code:

```
    n = 10
        a = 2.0082484079e+00
        b = -5.5511151231e-17
        c = 1.0010288241e+00
        d = 1.9989945632e+00

    n = 20
        a = 2.0020576483e+00
        b = -1.3877787808e-16
        c = 1.0002570672e+00
        d = 1.9997485744e+00

    n = 50
        a = 2.0003290247e+00
        b = 3.0531133177e-16
        c = 1.0000411245e+00
        d = 1.9999597689e+00
```

Problem 3

Wtie a function `midpoint(f, a, b, n)` *that implements the composite midpoint rule. Your function should take as input the name of the function to integrate,* `f`*, the endpoints of the interval of integration,* `a` *and* `b`*, and the number of points* `n` *of subintervals to divide the integral of integration into. Use the function to approximate the same integrals as in* **Problem 1** *with values of* `n` $= 10$*,* 20*, amd* 50*. In each case indicate the error.*

(a) $\displaystyle\int_0^\pi \sin x \; dx$

The error is bounded by $|E_S| \leq \dfrac{K(b-a)^5}{180n^4}$, where $a = 0$, $b = \pi$, and $K = \left|\displaystyle\max_{x\in(0,\pi)} \frac{d^4}{dx^4}\sin x\right| = 1$. So, $E_T \leq \dfrac{\pi^5}{180n^4}$

n	S_n	**Error Bound**	**Actual Error**
10	2.0001095173	$\frac{\pi^5}{180(10)^4} = 0.000170011$	0.0001095173
20	2.0000067844	$\frac{\pi^5}{180(20)^4} = 1.06256835 \times 10^{-5}$	6.7844×10^{-6}
50	2.0000001733	$\frac{\pi^5}{180(50)^4} = 2.720174976 \times 10^{-7}$	1.733×10^{-7}
100	2.0000000108	$\frac{\pi^5}{180(100)^4} = 1.70010936 \times 10^{-8}$	1.08×10^{-8}
200	2.0000000007	$\frac{\pi^5}{180(200)^4} = 1.06256835 \times 10^{-9}$	7×10^{-10}

(b) $\displaystyle\int_0^\pi \cos x \; dx$

The error is bounded by $|E_S| \leq \dfrac{K(b-a)^5}{180n^4}$, where $a = 0$, $b = \pi$, and $K = \left|\displaystyle\max_{x\in(0,\pi)} \frac{d^4}{dx^4}\cos x\right| = 1$. So, $E_S \leq \dfrac{\pi^5}{180n^4}$

n	S_n	**Error Bound**	**Actual Error**
10	0.0000000000	$\frac{\pi^5}{180(10)^4} = 0.000170011$	0.0000000000
20	$3.3306690739 \times 10^{-16}$	$\frac{\pi^5}{180(20)^4} = 1.06256835 \times 10^{-5}$	$3.3306690739 \times 10^{-16}$
50	$-8.3266726847 \times 10^{-17}$	$\frac{\pi^5}{180(50)^4} = 2.720174976 \times 10^{-7}$	$8.3266726847 \times 10^{-17}$
100	$-5.2735593670 \times 10^{-16}$	$\frac{\pi^5}{180(100)^4} = 1.70010936 \times 10^{-8}$	$5.2735593670 \times 10^{-16}$
200	$6.9388939039 \times 10^{-18}$	$\frac{\pi^5}{180(200)^4} = 1.06256835 \times 10^{-9}$	$6.9388939039 \times 10^{-18}$

(c) $\displaystyle\int_0^{\pi/2} \sin x \; dx$

The error is bounded by $|E_S| \leq \dfrac{K(b-a)^5}{180n^4}$, where $a = 0$, $b = \frac{\pi}{2}$, and
$K = \left| \max\limits_{x \in (0, \frac{\pi}{2})} \dfrac{d^4}{dx^4} \sin x \right| = 1.$ So, $E_S \leq \dfrac{\pi^5}{5760n^4}$

n	S_n	**Error Bound**	**Actual Error**
10	1.000003392220900	$\frac{\pi^5}{5760(10)^4} = 5.31284175 \times 10^{-6}$	$3.392220900 \times 10^{-6}$
20	1.000000211546591	$\frac{\pi^5}{5760(20)^4} = 3.320526094 \times 10^{-7}$	$2.11546591 \times 10^{-7}$
50	1.000000005412252	$\frac{\pi^5}{5760(50)^4} = 8.5005468 \times 10^{-9}$	5.412252×10^{-9}
100	1.000000000338236	$\frac{\pi^5}{5760(100)^4} = 5.3128417 \times 10^{-10}$	3.38236×10^{-10}
200	1.000000000021139	$\frac{\pi^5}{5760(200)^4} = 3.3205261 \times 10^{-11}$	2.1139×10^{-11}

(d) $\displaystyle\int_0^{\ln 3} e^x \; dx$

The error is bounded by $|E_S| \leq \dfrac{K(b-a)^5}{180n^4}$, where $a = 0$, $b = \ln 3$, and

$K = \left| \max\limits_{x \in (0, \ln 3)} \dfrac{d^4}{dx^4} e^x \right| = 3.$ So, $E_T \leq \dfrac{(\ln 3)^5}{60n^4}$

n	S_n	**Error Bound**	**Actual Error**
10	2.000001616261506	$\frac{(\ln 3)^5}{60(10)^4} = 2.667294764 \times 10^{-6}$	$1.616261506 \times 10^{-6}$
20	2.000000101125187	$\frac{(\ln 3)^5}{60(20)^4} = 1.667059228 \times 10^{-7}$	$1.01125187 \times 10^{-7}$
50	2.000000002589586	$\frac{(\ln 3)^5}{60(50)^4} = 4.267671623 \times 10^{-9}$	2.589586×10^{-9}
100	2.000000000161856	$\frac{(\ln 3)^5}{60(100)^4} = 2.6672948 \times 10^{-10}$	1.61856×10^{-10}
200	2.000000000010117	$\frac{(\ln 3)^5}{60(200)^4} = 1.6670592 \times 10^{-11}$	1.0117×10^{-11}

The following Python code was used to generate the above approximations:

```
from __future__ import division
from math import sin, cos, pi, exp, log

def rect_area(b, h):
    return b*h

def trap_area(b_1, b_2, h):
    return (1/2)*(b_1 + b_2)*h

def simp_rule(f, a, b):
    M = rect_area(f((a+b)/2), (b-a))
    T = trap_area(f(a), f(b), (b-a))
    return (2*M + T)/3

def comp_simp_rule(f, a, b, n):
    h = (b-a)/n

    approx = 0
    for i in xrange(0, int(n/2)):
        approx += simp_rule(f, a+(2*i*h), a+((2*i+2)*h))
    return approx

def problem_3():
    for n in [10, 20, 50, 100, 200]:
        print "\n    n = %d" % n
        a = comp_simp_rule(sin, 0, pi, n)
        b = comp_simp_rule(cos, 0, pi, n)
        c = comp_simp_rule(sin, 0, (pi/2), n)
        d = comp_simp_rule(exp, 0, log(3), n)
        print "        a = %.10e" % a
        print "        b = %.10e" % b
        print "        c = %.10e" % c
        print "        d = %.10e" % d
```

```python
problem_3()
```

Listing 3: Problem 3 source code

The following is the output of the above Python code:

```
    n = 10
        a = 2.0001095173e+00
        b = 0.0000000000e+00
        c = 1.000003392220900e+00
        d = 2.000001616261506e+00

    n = 20
        a = 2.0000067844e+00
        b = 3.3306690739e-16
        c = 1.000000211546591e+00
        d = 2.000000101125187e+00

    n = 50
        a = 2.0000001733e+00
        b = -8.3266726847e-17
        c = 1.000000005412252e+00
        d = 2.000000002589586e+00

    n = 100
        a = 2.0000000108e+00
        b = -5.2735593670e-16
        c = 1.000000000338236e+00
        d = 2.000000000161856e+00

    n = 200
        a = 2.0000000007e+00
        b = 6.9388939039e-18
        c = 1.000000000021139e+00
        d = 2.000000000010117e+00
```

Problem 4

Write a function `Dfwd(f, x, h=1e-6)` *that implements the 1st order finite difference approximation of the derivative*

$$f'(x) \approx \frac{f(x+h) - f(x)}{h}$$

Use this formula to approximate the derivative of the following functions at the specified values of x for $h = 0.1$, 0.05, 0.01, and 0.001 and calculate the error and the error bound. Present your results for each function in a table like the one for the revious problems:

(a) $f(x) = e^x$ at $x = 0$. The exact value is

$$f'(x) = e^x$$
$$f'(0) = e^0 = 1$$

The error is bounded by $|E| \leq \dfrac{hK}{2}$, where $K = \max_{x \in (0,h)} |f''(x)|$. Since $f''(x) = e^x$ is an increasing positive function, $K = e^h$. So, $E \leq \dfrac{he^h}{2}$

h	$\sim f'(0)$	**Error Bound**	**Actual Error**
0.1	1.0517091808	$\frac{0.1e^{0.1}}{2} = 0.0552585$	0.0517091808
0.05	1.0254219275	$\frac{0.051e^{0.05}}{2} = 0.026281777$	0.0254219275
0.01	1.0050167084	$\frac{0.01e^{0.01}}{2} = 0.00505025$	0.0050167084
0.001	1.0005001667	$\frac{0.001e^{0.001}}{2} = 0.000500500$	0.0005001667

(b) $f(x) = e^{-2x^2}$ at $x = 0$. The exact value is

$$f'(x) = -4x\ e^{-2x^2}$$
$$f'(0) = -4(0)\ e^0 = 0$$

The error is bounded by $|E| \leq \dfrac{hK}{2}$, where $K = \max_{x \in (0,h)} |f''(x)|$. Since $f''(x) = 4e^{-2x^2}(4x^2 - 1)$ is an increasing negative function on $(0, h)$ (for the h's we are considering), $K = |f''(0)| = 4$. So, $E \leq 2h$

h	$\sim f'(0)$	**Error Bound**	**Actual Error**
0.1	-0.19801326693	$2(0.1) = 0.2$	0.19801326693
0.05	-0.099750416146	$2(0.05) = 0.1$	0.099750416146
0.01	-0.019998000133	$2(0.01) = 0.02$	0.019998000133
0.001	-0.0019999980000	$2(0.001) = 0.002$	0.0019999980000

(c) $f(x) = \cos x$ at $x = 2\pi$. The exact value is

$$f'(x) = -\sin x$$
$$f'(0) = -\sin 0 = 0$$

The error is bounded by $|E| \leq \dfrac{hK}{2}$, where $K = \max\limits_{x\in(2\pi,2\pi+h)} |f''(x)|$. Since $f''(x) = -\cos x$ is an increasing negative function on $(2\pi, 2\pi + h)$ (for the h's we are considering), $K = |f''(0)| = 1$. So, $E \leq \dfrac{h}{2}$

h	$\sim f'(0)$	**Error Bound**	**Actual Error**
0.1	-0.049958347220	$\frac{0.1}{2} = 0.05$	0.049958347220
0.05	-0.024994792101	$\frac{0.05}{2} = 0.025$	0.024994792101
0.01	-0.0049999583335	$\frac{0.01}{2} = 0.005$	0.0049999583335
0.001	-0.00049999995833	$\frac{0.001}{2} = 0.0005$	0.00049999995833

(d) $f(x) = \ln x$ at $x = 1$. The exact value is

$$f'(x) = \frac{1}{x}$$
$$f'(1) = \frac{1}{1} = 1$$

The error is bounded by $|E| \leq \dfrac{hK}{2}$, where $K = \max\limits_{x\in(1,1+h)} |f''(x)|$. Since $f''(x) = -\dfrac{1}{x^2}$ is an increasing negative function on $(1, 1 + h)$, $K = |f''(1)| = 1$. So, $E \leq \dfrac{h}{2}$

h	$\sim f'(0)$	**Error Bound**	**Actual Error**
0.1	0.95310179804	$\frac{0.1}{2} = 0.05$	0.04689820196
0.05	0.97580328339	$\frac{0.05}{2} = 0.025$	0.02419671661
0.01	0.99503308532	$\frac{0.01}{2} = 0.005$	0.00496691468
0.001	0.99950033308	$\frac{0.001}{2} = 0.0005$	0.00049966692

The following Python code was used to generate the above approximations:

```python
from __future__ import division
from math import cos, pi, exp, log

def first_order_deriv_approx(f, x, h=1e-6):
    return (f(x+h) - f(x))/h

def exp_4b(x):
    return exp(-2*(x**2))

def problem_4():
    for h in [0.1, 0.05, 0.01, 0.001]:
        print "\n    h = %.3f" % h
        a = first_order_deriv_approx(exp, 0, h)
        b = first_order_deriv_approx(exp_4b, 0, h)
        c = first_order_deriv_approx(cos, 2*pi, h)
        d = first_order_deriv_approx(log, 1, h)
        print "        a = %.10e" % a
        print "        b = %.10e" % b
        print "        c = %.10e" % c
        print "        d = %.10e" % d

problem_4()
```

Listing 4: Problem 3 source code

The following is the output of the above Python code:

```
    h = 0.100
        a = 1.0517091808e+00
        b = -1.9801326693e-01
        c = -4.9958347220e-02
        d = 9.5310179804e-01

    h = 0.050
        a = 1.0254219275e+00
        b = -9.9750416146e-02
        c = -2.4994792101e-02
```

```
        d = 9.7580328339e-01

    h = 0.010
        a = 1.0050167084e+00
        b = -1.9998000133e-02
        c = -4.9999583335e-03
        d = 9.9503308532e-01

    h = 0.001
        a = 1.0005001667e+00
        b = -1.9999980000e-03
        c = -4.9999995833e-04
        d = 9.9950033308e-01
```

Problem 5

Repeat **Problem 4** *for the 2nd order centered difference*

$$f'(x) \approx \frac{f(x+h) - f(x-h)}{2h}$$

(a) $f(x) = e^x$ at $x = 0$. The error is bounded by $|E| \leq \dfrac{h^2 K}{6}$, where $K = \max\limits_{x \in (-h,h)} |f'''(x)|$. Since $f'''(x) = e^x$ is an increasing positive function, $K = e^h$. So, $E \leq \dfrac{h^2 e^h}{6}$

h	$\sim f'(0)$	**Error Bound**	**Actual Error**
0.1	1.001667500198441	$\frac{0.1^2 e^{0.1}}{6} = 0.00184195$	0.001667500198441
0.05	1.000416718753101	$\frac{0.051^2 e^{0.05}}{6} = 0.0004380296$	0.000416718753101
0.01	1.000016666749992	$\frac{0.01^2 e^{0.01}}{6} = 1.6834169 \times 10^{-5}$	$1.6666749992 \times 10^{-5}$
0.001	1.000000166666681	$\frac{0.001^2 e^{0.001}}{6} = 1.668334 \times 10^{-7}$	$1.66666681 \times 10^{-7}$

(b) $f(x) = e^{-2x^2}$ at $x = 0$. The error is bounded by $|E| \leq \dfrac{h^2 K}{6}$, where $K = \max\limits_{x \in (-h,h)} |f'''(x)|$. Since $f'''(x) = -16xe^{-2x^2}(4x^2 - 3)$ is an increasing positive function on $(0, h)$ (for the h's we are considering) and $f'''(x)$ is odd, $K = |f'''(h)|$. So, $E \leq \dfrac{-8h^3(4h^2-3)e^{-2h^2}}{3}$

h	$\sim f'(0)$	**Error Bound**	**Actual Error**
0.1	0.0000000000	$\frac{-8(0.1)^3(4(0.1)^2-3)e^{-2(0.1)^2}}{3} = 0.00773703486$	0.000000000000000
0.05	0.0000000000	$\frac{-8(0.05)^3(4(0.05)^2-3)e^{-2(0.05)^2}}{3} = 0.40435377$	0.000000000000000
0.01	0.0000000000	$\frac{-8(0.01)^3(4(0.01)^2-3)e^{-2(0.01)^2}}{3} = 7.9973337 \times 10^{-6}$	0.000000000000000
0.001	0.0000000000	$\frac{-8(0.001)^3(4(0.001)^2-3)e^{-2(0.001)^2}}{3} = 7.99997 \times 10^{-9}$	0.000000000000000

(c) $f(x) = \cos x$ at $x = 2\pi$. The error is bounded by $|E| \leq \dfrac{h^2 K}{6}$, where $K = \max\limits_{x \in (2\pi - h, 2\pi + h)} |f'''(x)|$. Since $f'''(x) = \sin x$ is an increasing negative function on $(2\pi, 2\pi + h)$ (for the h's we are considering) and $f'''(x)$ is odd and 2π-periodic, $K = |f'''(2\pi + h)| = sin(h)$. So, $E \leq \dfrac{h^2 \sin h}{6}$

h	$\sim f'(0)$	**Error Bound**	**Actual Error**
0.1	0.000000000000000	$\frac{(0.1)^2 \sin(0.1)}{6} = 0.000166389$	0.000000000000000
0.05	0.000000000000000	$\frac{(0.05)^2 \sin(0.05)}{6} = 2.082465386 \times 10^{-5}$	0.000000000000000
0.01	0.000000000000000	$\frac{(0.01)^2 \sin(0.01)}{6} = 1.666638889 \times 10^{-7}$	0.000000000000000
0.001	0.000000000000000	$\frac{(0.001)^2 \sin(0.001)}{6} = 1.666666 \times 10^{-10}$	0.000000000000000

(d) $f(x) = \ln x$ as $x = 1$. The error is bounded by $|E| \leq \dfrac{h^2 K}{6}$, where $K = \max_{x \in (1-h, 1+h)} |f'''(x)|$. Since $f'''(x) = \dfrac{2}{x^3}$ is a decreasing positive function on $(1-h, 1+h)$ (for the h's we are considering), $K = |f'''(1-h)|$. So, $E \leq \dfrac{h^2}{3(1-h)^3}$

h	$\sim f'(0)$	**Error Bound**	**Actual Error**
0.1	1.003353477310756	$\frac{(0.1)^2}{3(1-(0.1))^3} = 0.00457247$	0.003353477310756
0.05	1.000834585569826	$\frac{(0.05)^2}{3(1-(0.05))^3} = 0.00097195898$	0.000834585569826
0.01	1.000033335333477	$\frac{(0.01)^2}{3(1-(0.01))^3} = 3.43536717 \times 10^{-5}$	$3.3335333477 \times 10^{-5}$
0.001	1.000000333333479	$\frac{(0.001)^2}{3(1-(0.001))^3} = 3.3433534 \times 10^{-7}$	$3.33333479 \times 10^{-7}$

The following Python code was used to generate the above approximations:

```
from __future__ import division
from math import cos, pi, exp, log

def second_order_deriv_approx(f, x, h=1e-6):
    return (f(x+h) - f(x-h))/(2*h)

def exp_4b(x):
    return exp(-2*(x**2))

def problem_5():
    for h in [0.1, 0.05, 0.01, 0.001]:
        print "\n     h = %.3f" % h
        a = second_order_deriv_approx(exp, 0, h)
        b = second_order_deriv_approx(exp_4b, 0, h)
        c = second_order_deriv_approx(cos, 2*pi, h)
        d = second_order_deriv_approx(log, 1, h)
        print "         a = %.15e" % a
```

```
        print "        b = %.15e" % b
        print "        c = %.15e" % c
        print "        d = %.15e" % d

problem_5()
```

Listing 5: Problem 3 source code

The following is the output of the above Python code:

```
    h = 0.100
        a = 1.001667500198441e+00
        b = 0.000000000000000e+00
        c = 0.000000000000000e+00
        d = 1.003353477310756e+00

    h = 0.050
        a = 1.000416718753101e+00
        b = 0.000000000000000e+00
        c = 0.000000000000000e+00
        d = 1.000834585569826e+00

    h = 0.010
        a = 1.000016666749992e+00
        b = 0.000000000000000e+00
        c = 0.000000000000000e+00
        d = 1.000033335333477e+00

    h = 0.001
        a = 1.000000166666681e+00
        b = 0.000000000000000e+00
        c = 0.000000000000000e+00
        d = 1.000000333333479e+00
```

Problem 6

A particle moves along a trajectory in the xy plane such that at time t the object has position $(x(t), y(t))$. The velocity vector of the particle can be approximated by

$$v(t) \approx \left(\frac{x(t+\Delta t) - x(t)}{\Delta t^2}, \frac{y(t+\Delta t) - y(t)}{\Delta t^2} \right)$$

and its acceleration by

$$a(t) \approx \left(\frac{x(t+\Delta t) - 2x(t) + x(t-\Delta t)}{\Delta t^2}, \frac{y(t+\Delta t) - 2y(t) + y(t-\Delta t)}{\Delta t^2} \right)$$

Write a function `kinematics(x, y, t, dt=1.0e-6)` *that approximates the velocity and acceleration of a particle with position vector* $(\cos 2\pi t, \sin 2\pi t)$ *for* `t` $= 0, \frac{1}{4}, \frac{1}{2}, \frac{3}{4}$*, and* `dt` $= 0.1$*, 0.05, and* 0.01*. Calculate the error in the approximations. Display your results in a table.*

The actual solutions are:

$$\begin{aligned}
v(t) = x'(t) &= (-2\pi \sin(2\pi t), 2\pi \cos(2\pi t)) \\
\implies v(0) &= (0, 2\pi) \\
v(\tfrac{1}{4}) &= (-2\pi, 0) \\
v(\tfrac{1}{2}) &= (0, -2\pi) \\
v(\tfrac{3}{4}) &= (2\pi, 0) \\
a(t) = v'(t) &= (-4\pi^2 \cos(2\pi t), -4\pi^2 \sin(2\pi t)) \\
\implies a(0) &= (-4\pi^2, 0) \\
a(\tfrac{1}{4}) &= (0, -4\pi^2) \\
a(\tfrac{1}{2}) &= (4\pi^2, 0) \\
a(\tfrac{3}{4}) &= (0, 4\pi^2)
\end{aligned}$$

`t`	`dt`	**Approximate Velocity**	**Velocity Error**	**Velocity Error Magnitude**
0	0.1	$(-1.90983, 5.87785)$	$(1.90983, 0.4053353)$	1.95236967
	0.05	$(-0.978870, 6.18034)$	$(0.978870, 0.1028453)$	0.9842579
	0.01	$(-0.197327, 6.27905)$	$(0.197327, 0.0041353)$	0.1973703
$\frac{1}{4}$	0.1	$(-5.87785, -1.90983)$	$(0.4053353, 1.90983)$	1.95236967
	0.05	$(-6.18034, -0.978870)$	$(0.1028453, 0.978870)$	0.9842579
	0.01	$(-6.27905, -0.197327)$	$(0.0041353, 0.197327)$	0.1973703
$\frac{1}{2}$	0.1	$(1.90983, -5.87785)$	$(1.90983, 0.4053353)$	1.95236967
	0.05	$(0.978870, -6.180340)$	$(0.978870, 0.1028453)$	0.9842579
	0.01	$(0.197327, -6.27905)$	$(0.197327, 0.0041353)$	0.1973703
$\frac{3}{4}$	0.1	$(5.87785, 1.90983)$	$(0.4053353, 1.90983)$	1.95236967
	0.05	$(6.180340, 0.978870)$	$(0.1028453, 0.978870)$	0.9842579
	0.01	$(6.27905, 0.197327)$	$(0.0041353, 0.197327)$	0.1973703

t	dt	Approximate Acceleration	Acceleration Error	Acceleration Error Magnitude
0	0.1	$(-38.196601, 0.0000000)$	$(1.281816, 0.0000000)$	1.281816
	0.05	$(-39.154787, 0.0000000$	$(0.3236306, 0.0000000)$	0.3236306
	0.01	$(-39.465431, 0.0000000)$	$(0.0129866, 0.0000000)$	0.0129866
$\frac{1}{4}$	0.1	$(1.11 \times 10^{-14}, -38.196601)$	$(1.11 \times 10^{-14}, 1.281816)$	1.281816
	0.05	$(0.0000000000, -39.154787$	$(0.0000000, 0.3236306)$	0.3236306
	0.01	$(-1.39 \times 10^{-13}, -39.465431)$	$(-1.39 \times 10^{-13}, 0.0129866)$	0.0129866
$\frac{1}{2}$	0.1	$(38.196601, 0.0000000)$	$(1.281816, 0.0000000)$	1.281816
	0.05	$(39.154787, -1.78 \times 10^{-13})$	$(0.3236306, -1.78 \times 10^{-13})$	0.3236306
	0.01	$(39.465431, 0.0000000))$	$(0.0129866, 0.0000000)$	0.0129866
$\frac{3}{4}$	0.1	$(0.0000000, 38.196601)$	$(0.0000000, 1.281816)$	1.281816
	0.05	$(2.22 \times 10^{-14}, 39.154787)$	$(2.22 \times 10^{-14}, 0.3236306)$	0.3236306
	0.01	$(0.0000000, 39.465431)$	$(0.0000000, 0.0129866)$	0.0129866

The following Python code was used to generate the above approximations:

```
from __future__ import division
from math import cos, pi, exp, log

def first_order_deriv_approx(f, x, h=1e-6):
    return (f(x+h) - f(x))/h

def kin_cos(t):
    return cos(2*pi*t)

def kin_sin(t):
    return sin(2*pi*t)

def kinematics(x, y, t, dt=1.0e-6):
    x_vel = first_order_deriv_approx(x, t, dt)
    y_vel = first_order_deriv_approx(y, t, dt)
    vel = (x_vel, y_vel)

    x_acc = (x(t+dt) - 2*x(t) + x(t-dt))/(dt**2)
```

```python
    y_acc = (y(t+dt) - 2*y(t) + y(t-dt))/(dt**2)
    acc = (x_acc, y_acc)

    return (vel, acc)
def problem_6():
    for t in [0, (1/4), (1/2), (3/4)]:
        print "\n    t = %.2f" % t
        for dt in [0.1, 0.05, 0.01]:
            print "\n        dt = %.2f" % dt
            (vel, acc) = kinematics(kin_cos, kin_sin, t, dt)
            print "            vel = (%.10e, %.10e)" % (vel[0], vel[1])
            print "            acc = (%.10e, %.10e)" % (acc[0], acc[1])

problem_6()
```

Listing 6: Problem 3 source code

The following is the output of the above Python code:

```
    t = 0.00

        dt = 0.10
            vel = (-1.9098300563e+00, 5.8778525229e+00)
            acc = (-3.8196601125e+01, 0.0000000000e+00)

        dt = 0.05
            vel = (-9.7886967410e-01, 6.1803398875e+00)
            acc = (-3.9154786964e+01, 0.0000000000e+00)

        dt = 0.01
            vel = (-1.9732715717e-01, 6.2790519529e+00)
            acc = (-3.9465431435e+01, 0.0000000000e+00)

    t = 0.25

        dt = 0.10
            vel = (-5.8778525229e+00, -1.9098300563e+00)
            acc = (1.1102230246e-14, -3.8196601125e+01)

        dt = 0.05
            vel = (-6.1803398875e+00, -9.7886967410e-01)
            acc = (0.0000000000e+00, -3.9154786964e+01)

        dt = 0.01
            vel = (-6.2790519529e+00, -1.9732715717e-01)
            acc = (-1.3877787808e-13, -3.9465431435e+01)

    t = 0.50

```

```
        dt = 0.10
            vel = (1.9098300563e+00, -5.8778525229e+00)
            acc = (3.8196601125e+01, 0.0000000000e+00)

        dt = 0.05
            vel = (9.7886967410e-01, -6.1803398875e+00)
            acc = (3.9154786964e+01, -1.7763568394e-13)

        dt = 0.01
            vel = (1.9732715717e-01, -6.2790519529e+00)
            acc = (3.9465431435e+01, 0.0000000000e+00)

    t = 0.75

        dt = 0.10
            vel = (5.8778525229e+00, 1.9098300563e+00)
            acc = (0.0000000000e+00, 3.8196601125e+01)

        dt = 0.05
            vel = (6.1803398875e+00, 9.7886967410e-01)
            acc = (2.2204460493e-14, 3.9154786964e+01)

        dt = 0.01
            vel = (6.2790519529e+00, 1.9732715717e-01)
            acc = (0.0000000000e+00, 3.9465431435e+01)
```