Answers to Homework 8: Numerical Differentiation

---

1. Approximate $f'(x_0)$ and $f''(x_0)$ using the values of $f(x)$ at x_0-h, x_0 and $x_0+\alpha h$ ($\alpha > 0$)

   (a) applying the polynomial interpolation method

   (b) applying the Taylor series method

   Assuming $f(x) \in C^3$, evaluate the approximation error using either of the two methods. What is the approximation order?

   Answer:

   (a) The interpolation polynomial in the Lagrange form is

$$\begin{aligned} P(x) &= f(x_0-h)\frac{(x-x_0)(x-x_0-\alpha h)}{(x_0-h-x_0)(x_0-h-x_0-\alpha h)} \\ &+ f(x_0)\frac{(x-x_0+h)(x-x_0-\alpha h)}{(x_0-x_0+h)(x_0-x_0-\alpha h)} \\ &+ f(x_0+\alpha h)\frac{(x-x_0)(x-x_0+h)}{(x_0+\alpha h-x_0)(x_0+\alpha h-x_0+h)} \\ &= f(x_0-h)\frac{(x-x_0)(x-x_0-\alpha h)}{(1+\alpha)h^2} \\ &- f(x_0)\frac{(x-x_0+h)(x-x_0-\alpha h)}{\alpha h^2} \\ &+ f(x_0+\alpha h)\frac{(x-x_0)(x-x_0+h)}{\alpha(1+\alpha)h^2} \end{aligned}$$

   Evaluating its first derivative at x_0, we get

$$\begin{aligned} f'(x_0) \approx P'(x_0) &= f(x_0-h)\frac{-\alpha h}{(1+\alpha)h^2} - f(x_0)\frac{h-\alpha h}{\alpha h^2} + f(x_0+\alpha h)\frac{h}{\alpha(1+\alpha)h^2} \\ &= \frac{f(x_0+\alpha h)+(\alpha^2-1)\,f(x_0)-\alpha^2\,f(x_0-h)}{\alpha(1+\alpha)h} \end{aligned}$$

   For $\alpha = 1$, this formula reduces to the central difference approximation.
   Evaluating the second derivative of $P(x)$, we get

$$\begin{aligned} f'(x_0) \approx P''(x_0) &= f(x_0-h)\frac{2}{(1+\alpha)h^2} - f(x_0)\frac{2}{\alpha h^2} + f(x_0+\alpha h)\frac{2}{\alpha(1+\alpha)h^2} \\ &= \frac{2\left[f(x_0+\alpha h)-(\alpha+1)\,f(x_0)+\alpha\,f(x_0-h)\right]}{\alpha(1+\alpha)h^2} \end{aligned}$$

   (b) The Taylor series expressions for $f(x_0-h)$ and $f(x_0+\alpha h)$ are

$$\begin{aligned} f(x_0-h) &= f(x_0)-h\,f'(x_0)+\frac{h^2}{2}\,f''(x_0)-\frac{h^3}{6}\,f'''(\xi_1)\,, \\ f(x_0+\alpha h) &= f(x_0)+\alpha h\,f'(x_0)+\frac{\alpha^2 h^2}{2}\,f''(x_0)+\frac{\alpha^3 h^3}{6}\,f'''(\xi_1)\,, \end{aligned}$$

where ξ_1 is a point between $x_0 - h$ and x_0 and ξ_2 is a point between x_0 and $x_0 + \alpha h$. Looking for an approximation for the first derivative of the form

$$f'(x_0) \approx A_1\, f(x_0 + \alpha h) + B_1\, f(x_0) + C_1\, f(x_0 - h)\,,$$

we arrive at the system of three linear equations to determine the weights A, B, and C:

$$\begin{aligned} A_1 + B_1 + C_1 &= 0\,, \\ h\,(\alpha\, A_1 - C_1) &= 1\,, \\ \frac{h^2}{2}\,(\alpha^2\, A_1 + C_1) &= 0\,. \end{aligned}$$

From the third equation we find

$$C_1 = -\alpha^2\, A_1\,.$$

Substituting this into the second equation we find

$$A_1 = \frac{1}{\alpha\,(1+\alpha)\,h}\,.$$

Finally, from the first equation, we find

$$B_1 = (\alpha^2 - 1)\, A_1 = \frac{\alpha - 1}{\alpha\, h}\,.$$

Putting it together,

$$f'(x_0) \approx \frac{f(x_0 + \alpha h) + (\alpha^2 - 1)\, f(x_0) - \alpha^2\, f(x_0 - h)}{\alpha\,(1+\alpha)\,h}$$

Looking for an approximation for the second derivative of the form

$$f''(x_0) \approx A_2\, f(x_0 + \alpha h) + B_2\, f(x_0) + C_2\, f(x_0 - h)\,,$$

we arrive at the analogous system of three linear equations to determine the weights A, B, and C:

$$\begin{aligned} A_2 + B_2 + C_2 &= 0\,, \\ h\,(\alpha\, A_2 - C_2) &= 0\,, \\ \frac{h^2}{2}\,(\alpha^2\, A_2 + C_2) &= 1\,. \end{aligned}$$

From the second equation we find

$$C_2 = \alpha\, A_2\,.$$

Substituting this into the third equation we find

$$A_2 = \frac{2}{\alpha\,(1+\alpha)\,h^2}\,.$$

Finally, from the first equation, we find

$$B_2 = -(\alpha + 1)\, A_2 = -\frac{2}{\alpha\, h}\,.$$

Putting it together,

$$f'(x_0) \approx \frac{f(x_0 + \alpha h) - (\alpha + 1)\, f(x_0) + \alpha\, f(x_0 - h)}{\alpha\,(1+\alpha)\,h^2}$$

We can find the error of these approximations, for example, from the Taylor series method.

The error of the first derivative approximation

$$\begin{aligned} E_1 &= f'(x_0) - \frac{f(x_0+\alpha h) + (\alpha^2-1)\, f(x_0) - \alpha^2\, f(x_0-h)}{\alpha(1+\alpha)h} \\ &= -\frac{h^3}{6}\left[\alpha^3 A_1\, f'''(\xi_2) - C_1\, f'''(\xi_1)\right] = -\frac{\alpha h^2}{6(1+\alpha)}\left[\alpha\, f'''(\xi_2) + f'''(\xi_1)\right] \end{aligned}$$

The expression in the square brackets is between $f'''(\xi_1)(\alpha+1)$ and $f'''(\xi_2)(\alpha+1)$. Since $f'''(x)$ is a continuous function, we can apply the intermediate value theorem and replace this expression with $f'''(\xi)(\alpha+1)$, where ξ is some point between ξ_1 and ξ_2. Finally,

$$E_1 = -\frac{\alpha h^2}{6}\, f'''(\xi) = O(h^2)$$

The approximation has second order. For $\alpha < 1$, the error can be smaller than the error of the central difference approximation.

The error of the second derivative approximation

$$\begin{aligned} E_2 &= f''(x_0) - \frac{f(x_0+\alpha h) - (\alpha+1)\, f(x_0) + \alpha\, f(x_0-h)}{\alpha(1+\alpha)h^2} \\ &= -\frac{h^3}{6}\left[\alpha^3 A\, f'''(\xi_2) - C\, f'''(\xi_1)\right] = -\frac{h}{3(1+\alpha)}\left[\alpha^2\, f'''(\xi_2) - f'''(\xi_1)\right] \end{aligned}$$

Using the same argument as before, we can reduce this to

$$E_2 = \frac{(1-\alpha)h}{3}\, f'''(\xi) = O(h)\,,$$

where ξ is some point between $x_0 - h$ and $x_0 + \alpha h$. The approximation is only first order (unless $\alpha = 1$).

2. Apply the polynomial interpolation method at $2n+1$ regularly spaced points

$$x_{-n}, x_{-n+1}, \ldots, x_{-1}, x_0, x_1, \ldots, x_n$$

with $x_k = x_0 + k\,h$, $k = -n, -n+1, \ldots, -1, 0, 1, \ldots, n-1, n$ to derive the approximation

$$f^{(2n)}(x_0) \approx \frac{\Delta^{2n} f(x_{-n})}{h^{2n}} = \sum_{k=-n}^{n} \frac{(-1)^{k+n}}{h^{2n}} \binom{2n}{k+n} f(x_k), \tag{1}$$

where $\Delta f(x) = f(x+h) - f(x)$.

Hint: Recall Stirling's interpolation formula from Homework 5.

Answer:

Stirling's interpolation formula has the form

$$\begin{aligned} P(x) \;=\; & f(x_0) + s\,\frac{\Delta f(x_{-1}) + \Delta f(x_0)}{2} + \frac{s^2}{2}\,\Delta^2 f(x_{-1}) + \\ & \frac{s\left(s^2-1^2\right)}{3!}\,\frac{\Delta^3 f(x_{-2}) + \Delta^3 f(x_{-1})}{2} + \frac{s^2\left(s^2-1^2\right)}{4!}\,\Delta^4 f(x_{-2}) + \ldots + \\ & \frac{s\left(s^2-1\right)\left(s^2-2^2\right)\ldots\left(s^2-(n-1)^2\right)}{(2n-1)!}\,\frac{\Delta^{2n-1} f(x_{-n}) + \Delta^{2n-1} f(x_{-n+1})}{2} + \\ & \frac{s^2\left(s^2-1\right)\left(s^2-2^2\right)\ldots\left(s^2-(n-1)^2\right)}{(2n)!}\,\Delta^{2n} f(x_{-n}), \end{aligned}$$

where $s = (x - x_0)/h$. Extracting the leading-order polynomial coefficient, we have

$$P(x) = \frac{s^{2n}}{(2n)!}\,\Delta^{2n} f(x_{-n}) + \ldots$$

where the omitted terms contain smaller powers of s. Differentiating this expression $2n$ times and noting that

$$\frac{d}{dx} = \frac{1}{h}\frac{d}{ds},$$

we get

$$P^{(2n)}(x) = \frac{1}{h^{2n}}\,\frac{(2n)!}{(2n)!}\,\Delta^{2n} f(x_{-n}) = \frac{\Delta^{2n} f(x_{-n})}{h^{2n}}.$$

This is the required approximation for $f^{(2n)}(x_0)$.

Note: The second equality follows from the general equation

$$\Delta^n f(x_m) = \sum_{k=m}^{m+n} (-1)^{k+n+m} \binom{n}{k-m} f(x_k)$$

which is proved by the method of mathematical induction. First, check that the equation is true for $n = 1$:

$$\Delta f(x_m) = f(x_{m+1}) - f(x_m) = \sum_{k=m}^{m+1} (-1)^{k+m+1} \binom{1}{k-m} f(x_k)$$

Next, let us suppose that the general equation is true for some n and prove it for $n+1$. We get

$$\Delta^{n+1} f(x_m) = \Delta\Delta^n f(x_m) = \sum_{k=m}^{m+n} (-1)^{k+n+m} \binom{n}{k-m} f(x_{k+1}) - \sum_{k=m}^{m+n} (-1)^{k+n+m} \binom{n}{k-m} f(x_k).$$

Shifting the index in the first sum leads to

$$\begin{aligned}\Delta^{n+1} f(x_m) &= \sum_{k=m+1}^{m+n+1} (-1)^{k+n+m-1} \binom{n}{k-m-1} f(x_k) - \sum_{k=m}^{m+n} (-1)^{k+n+m} \binom{n}{k-m} f(x_k) \\ &= f(x_{m+n+1}) + \sum_{k=m+1}^{m+n} (-1)^{k+n+m-1} \left[\binom{n}{k-m-1} + \binom{n}{k-m} \right] f(x_k) - (-1)^n f(x_m).\end{aligned}$$

We can simplify the term in the square brackets since

$$\begin{aligned}\binom{n}{k-m-1} + \binom{n}{k-m} &= \frac{n!}{(k-m-1)!(n-k+m)!} \left(\frac{1}{n-k+m+1} + \frac{1}{k-m} \right) \\ &= \frac{(n+1)!}{(k-m)!(n-k+m+1)!} = \binom{n+1}{k-m}.\end{aligned}$$

Collecting the terms, we finally obtain

$$\Delta^{n+1} f(x_m) = \sum_{k=m}^{m+n+1} (-1)^{k+n+m+1} \binom{n+1}{k-m} f(x_k),$$

which completes the proof.

3. Richardson extrapolation can be implemented with the following algorithm:

```
RICHARDSON(N(x), h, tol, n)
 1  for k ← 1,2,...,n
 2  do
 3     R_{k,1} ← N(h)
 4     t ← 1
 5     for i ← 1,2,...,k−1
 6     do
 7        t ← 2t
 8        R_{k,i+1} ← R_{k,i} + (R_{k,i} − R_{k−1,i})/(t − 1)
 9     if k > 1 and |R_{k,k} − R_{k−1,k−1}| ≤ tol
10       then return R_{k,k}
11     h ← h/2
12  return R_{n,n}
```

The algorithm successively fills the rows of the triangular matrix

$$\begin{bmatrix} R_{1,1} & & & \\ R_{2,1} & R_{2,2} & & \\ \vdots & \vdots & \ddots & \\ R_{n,1} & R_{n,2} & \cdots & R_{n,n} \end{bmatrix}. \tag{2}$$

Modify the algorithm so that only one row of length n is stored in memory instead of the whole matrix.

Hint: Rearrange the matrix in the form

$$\begin{bmatrix} & & & R_{1,1} \\ & & R_{2,1} & R_{2,2} \\ & \iddots & \vdots & \vdots \\ R_{n,1} & \cdots & R_{n,n-1} & R_{n,n} \end{bmatrix}. \tag{3}$$

Answer:

```
RICHARDSON2(N(x), h, tol, n)
 1  for k ← 1,2,...,n
 2  do
 3     R_{n-k+1} ← N(h)
 4     t ← 1
 5     for i ← n-k+1, n-k+2, ..., n-1
 6     do
 7        t ← 2t
 8        R_{i+1} ← R_i + (R_i - R_{i+1})/(t-1)
 9     if k > 1 and |R_n - R| ≤ tol
10       then return R_n
11     R ← R_n
12     h ← h/2
13  return R
```

4. (Programming) In Homework 1, we applied an ancient geometric method to compute the value of π. The approximation formula is

$$\pi \approx \frac{k L_k}{2}, \tag{4}$$

where L_k (the side of a regular polygon) satisfies the recursion

$$L_{2k} = \frac{L_k}{\sqrt{2+\sqrt{4-L_k^2}}}. \tag{5}$$

starting with $L_6 = 1$.

Implement the Richardson extrapolation algorithm and apply it to accelerate the convergence of the geometric estimation of π. Start with $k = 6$, take $h = 6/k$, $N(h) = k L_k/2$ and output five rows of the Richardson table.

Answer:

3.00000				
3.10583	3.21166			
3.13263	3.15943	3.14202		
3.13935	3.14607	3.14162	3.14156	
3.14103	3.14271	3.14159	3.14159	3.14159

C program:

```
#include <stdio.h>  /* for output */
#include <stdlib.h> /* for allocation */
#include <math.h>   /* for math functions */
#include <assert.h> /* for assertion  */

/* Function: richardson
   --------------------
   Rihchardson extrapolation (algorithm 1)
   N(k) - numerical approximation
   tol  - accuracy tolerance
   n    - number of extrapolation levels
*/
double richardson (double (*N)(int), double tol, int n)
{
    int k, i;
    double t, r, **R;

    /* allocate space for the matrix rows */
    R = (double**) malloc (n*sizeof(double*));
    assert (R != NULL);

    /* loop over rows */
    for (k=0; k < n; k++) {
        /* allocate space: row by row */
R[k] = (double*) malloc ((k+1)*sizeof(double));
assert (R[k] != NULL);

R[k][0] = N(k);
printf ("%d \t %g",k,R[k][0]); /* start row */

for (i=0, t = 2; i < k; i++, t *= 2) {
```

```
    R[k][i+1] = R[k][i] + (R[k][i] - R[k-1][i])/(t-1.);
    printf("\t %g",R[k][i+1]); /* output row */
}
printf("\n"); /* end row */

r = R[k][k];
if (k > 0 && fabs(r - R[k-1][k-1]) <= tol) break;
    }

    /* free space */
    for (i=0; i < k; i++) {
free (R[i]);
    }
    if (k < n) free (R[k]);
    free (R);

    return r;
}

/* Function: richardson2
   ---------------------
   Rihchardson extrapolation (algorithm 2)
   N(k) - numerical approximation
   tol  - accuracy tolerance
   n    - number of extrapolation levels
*/
double richardson2 (double (*N)(int), double tol, int n)
{
    int k, i;
    double t, r, *R;

    /* allocate space: only one row of the table */
    R = (double*) malloc (n*sizeof(double));
    assert (R != NULL);

    /* loop over rows */
    for (k=0; k < n; k++) {
        R[n-k-1] = N(k);
        printf ("%d \t %g",k,R[n-k-1]); /* start row */

        /* loop over columns */
        for (i=n-k-1, t = 2; i < n-1; i++, t *= 2) {
            R[i+1] = R[i] + (R[i] - R[i+1])/(t-1.);
            printf("\t %g",R[i+1]); /* output row */
        }
        printf("\n"); /* end row */

        if (k > 0 && fabs(r - R[n-1]) <= tol) break;
        r = R[n-1];
    }
    free (R);

    if (k==n)
        fprintf(stderr,"Tolerance %g not reached after %d levels\n",tol,n);

    return r;
}

/* Function: polygon
```

```
   -----------------
   Computes the geometric approximation of pi
   k - approximation level
*/
double polygon (int k)
{
    static double l2=1.; /* polygon side squared */
    static int n=3;      /* number of sides / 2  */

    if (k > 0) {
        n *= 2;
        l2 = l2/(2.0 + sqrt(4.0-l2));
    }

    return (n*sqrt(l2));
}

/* Main program */
int main (void) {
    double pi;

    pi = richardson2 (polygon, 0., 5);

    exit (0);
}
```

5. (Programming)

(a) Compute the derivative of $f(x) = \sin x$ at $x = \frac{\pi}{3}$ using

i. the forward difference approximation

ii. the central difference approximation

iii. your approximation from problem 1 with $\alpha = 1/2$.

Use the step size $h = 10^{-n}$ for $n = 0, 1, 2, \ldots, 15$. Perform all the computations with double precision and output your results in a table. Explain the difference between the rows and columns of the table.

Answer:

n	Forward	Central	Skewed
0	0.022626	0.420735	0.451210
1	0.455902	0.499167	0.499574
2	0.495662	0.499992	0.499996
3	0.499567	0.500000	0.500000
4	0.499957	0.500000	0.500000
5	0.499996	0.500000	0.500000
6	0.500000	0.500000	0.500000
7	0.500000	0.500000	0.500000
8	0.500000	0.500000	0.500000
9	0.500000	0.500000	0.500000
10	0.500000	0.500000	0.500000
11	0.500000	0.500000	0.499999
12	0.500044	0.500044	0.500035
13	0.499600	0.499600	0.499504
14	0.499600	0.499600	0.506033
15	0.555112	0.555112	0.471411

Explanation: Both the central difference approximation and the skewed difference approximation from problem 1 converge faster to the exact value of 0.5 thanks to their second order. The convergence of the skewed approximation is somewhat faster because of the smaller coefficient in the error term. At very small steps h, all approximations diverge because of the round-off error.

(b) Compute the derivative of $f(x) = \sin x$ at $x = \frac{\pi}{3}$ using the forward difference approximation and the Richardson extrapolation algorithm. Start with $h = 1$ and find the number of rows in the Richardson table required to estimate the derivative with six significant decimal digits. Output the table.

Answer:

```
0.0226256
0.267392    0.512159
0.387117    0.506842    0.505070
0.444643    0.502168    0.500610    0.499973
0.472620    0.500597    0.500074    0.499997    0.499999
0.486388    0.500156    0.500009    0.500000    0.500000    0.500000
```

Six rows of the Richardson table are sufficient to estimate the derivative with six correct decimal digits.

C program:

```
#include <stdio.h>  /* for output */
#include <math.h>   /* for math functions */

/* see the implementation of Richardson in the previous problem */
#include "richardson.h"

/* Function: forward
   -----------------
   Forward-difference approximation to the first derivative of sin(x)
   k - approximation level
*/
double forward (int k)
{
    static double x0, f0, h=1.;
    double fp, diff;

    if (k == 0) {
        x0 = acos(-1.)/3.; /* pi/3 */
        f0 = sin(x0);
    } else {
        h /= 2; /* decrease the step */
    }

    fp = sin(x0+h);
    diff = (fp-f0)/h; /* forward difference */
    return diff;
}

int main (void)
{
    int i;
    double h, forw, cntr, skew, x0, f0, fp, fm;

    /* pi/3 */
    x0 = acos(-1.)/3.;
    f0 = sin(x0);
    for (i=0, h=1; i < 16; i++, h /= 10) {
        fp = sin(x0+h);
        fm = sin(x0-h);

        /* forward difference */
        forw = (fp-f0)/h;
        /* central difference */
        cntr = (fp-fm)/(2.*h);
        /* skewed difference, alpha=0.5 */
        skew = (sin(x0+0.5*h)-0.75*f0-0.25*fm)/(0.75*h);
        /* print table */
        printf("%d \t %f \t %f \t %f\n", i,forw,cntr,skew);
    }

    /* problem b */
    forw = richardson2 (forward, 1.e-6, 100);

    return 0;
}
```