STAT 812: Computational Statistics

Computer Arithmetic

Longhai Li

Contents


```
options(digits=22) ## display values in 22 digits
```

1 Integer

An integer u is represented by a vector of binary numbers $(d_0, \ldots, d_{31})$ as follows:

$$u = \sum_{i=0}^{31} d_i 2^i - 2^{31}.$$

The larguest integer is

$$u^{max} = \sum_{i=0}^{31} 1 \times 2^i - 2^{31} = 2^{31} - 1.$$

The smallest integer is

$$u^{min} = -2^{31} \approx -(2^{31} - 1).$$

Checking the largest and smallest integers

```
as.integer(2^31)
```

```
## Warning: NAs introduced by coercion to integer range
```

```
## [1] NA
```

```
as.integer(2^31 - 1)
```

```
## [1] 2147483647
```

```
as.integer(2^31 - 1) + as.integer(1)
```

```
## Warning in as.integer(2^31 - 1) + as.integer(1): NAs produced by integer
## overflow
```

```
## [1] NA
```

```
as.integer(-(2^31 - 1))
```

```
## [1] -2147483647
```

```
as.integer(- 2^31 )
```

```
## Warning: NAs introduced by coercion to integer range
```

```
## [1] NA
```

```
as.integer(- 2^31 -1 )
```

```
## Warning: NAs introduced by coercion to integer range
```

```
## [1] NA
```

2 Floating point numbers

A floating point number x is reprented by a vector of binary numbers $(s, d_0, \ldots, d_{t-1}, e_1, \ldots, e_{k-1})$ as follows:

$$x = (-1)^s \sum_{i=0}^{t-1} d_i 2^{-i} \times 2^{\sum_{i=0}^{k-1} e_i 2^i - 2^k}.$$

Typically, in double precision (64bits), $t = 52$ and $k = 11$. The largest positive floating point number is roughly 2^{1024} and the smallest positive floating point number is roughly 2^{-1074}. The numbers greater than 2^{1024} are overflow, called infinity. The numbers smaller than 2^{-1074} are underlfow, treated as 0.

```
# checking the largest double precision floating-point number:
2^1023
```

```
## [1] 8.988465674311579538647e+307
```

```
2^1023+1
```

```
## [1] 8.988465674311579538647e+307
```

```
2^1024
```

```
## [1] Inf
```

```
#smallest double precision floating-point number: > 2^(-1073)

2^(-1074)
```

```
## [1] 4.940656458412465441766e-324
```

```
2^(-1075)
```

```
## [1] 0
```

3 Rounding error

3.1 Computing sum: avoiding computing large + small

```
### demo of roundoff error: avoid large +- small
```

```r
# a number f0
f0 <- 1 + 2^(-40)

f0
```

```
## [1] 1.000000000000909494702
```

```r
f1 <- 1
i <- 1
while(i <= 2^12){
    f1 <- f1 + 2^(-52)
    i <- i + 1
}

f1
```

```
## [1] 1.000000000000909494702
```

```r
#A method that will give your wrong answer

f2 <- 1
i <- 1
while(i <= 2^13){
    f2 <- f2 + 2^(-53)
    i <- i + 1
}


f2
```

```
## [1] 1
```

```r
# this difference is not due to number display in R:
f0==f1
```

```
## [1] TRUE
```

```r
f0==f2
```

```
## [1] FALSE
```

```r
#R built-in function 'sum'

f3 <- sum (c(1, rep (2^(-53),2^13)))

f3
```

```
## [1] 1.000000000000909494702
```

```r
f1 == f3
```

```
## [1] TRUE
```

```r
f4 <- sum (c(1, rep (2^(-54),2^14)))

f4
```

```
## [1] 1.000000000000909494702
```

```r
f0 == f4
```

```
## [1] TRUE
```

```
# a possible solution is to sort the numbers before adding

f6s <- 0
i <- 1
while(i <= 2^14){
    f6s <- f6s + 2^(-54)
    i <- i + 1
}
print (f6s)
```

```
## [1] 9.094947017729282379e-13
```

```
f6 <- f6s + 1; f6
```

```
## [1] 1.000000000000909494702
```

Let's get a rough sense of mantissa in different bases:

```
#base= exp(1) = 2.71
log(2^(-52))
```

```
## [1] -36.04365338911715355152
```

```
1+ exp (-35)
```

```
## [1] 1.000000000000000666134
```

```
1+ exp (-37)
```

```
## [1] 1
```

```
#based 10
log(2^(-52),base=10)
```

```
## [1] -15.653559774527021630 77
```

3.2 Computing exp (x): avoiding computing large - large

```
fexp <- function(x, debug=FALSE)
{
    i <- 0
    expx <- 1
    u <- 1
    while(abs(u)>1e-20*abs(expx)) {
        i <- i+1
        u <- u*x/i
        expx <- expx+u
        if (debug){
            cat(sprintf ("Step %d: adding %f, exp(%f) = %f\n", i, u, x, expx))
        }
    }
    expx
}

matrix(c(exp(10),fexp(10)))
```

```
##                          [,1]
```

```
## [1,] 22026.46579480671789497
## [2,] 22026.46579480671061901
```

```
matrix(c(exp(20),fexp(20)))
```

```
##                          [,1]
## [1,] 485165195.4097902774811
## [2,] 485165195.4097902178764
```

```
matrix(c(exp(60),fexp(60)))
```

```
##                               [,1]
## [1,] 1142007389815684234540482 56
## [2,] 1142007389815684749936558 08
```

```
matrix(c(exp(-1),fexp(-1)))
```

```
##                            [,1]
## [1,] 0.3678794411714423340243
## [2,] 0.3678794411714424450466
```

```
matrix(c(exp(-10),fexp(-10)))
```

```
##                               [,1]
## [1,] 4.5399929762484854173 15e-05
## [2,] 4.5399929623031280972 32e-05
```

```
matrix(c(exp(-20),fexp(-20)))
```

```
##                               [,1]
## [1,] 2.0611536224385578699 42e-09
## [2,] 5.6218844721304176121 67e-09
```

```
matrix(c(exp(-50),fexp(-50)))
```

```
##                               [,1]
## [1,] 1.9287498479639178205 63e-22
## [2,] 1.1072933382891969813 50e+04
```

```
# See what's happening
fexp (-50, debug = TRUE)
```

```
## Step 1: adding -50.000000, exp(-50.000000) = -49.000000
## Step 2: adding 1250.000000, exp(-50.000000) = 1201.000000
## Step 3: adding -20833.333333, exp(-50.000000) = -19632.333333
## Step 4: adding 260416.666667, exp(-50.000000) = 240784.333333
## Step 5: adding -2604166.666667, exp(-50.000000) = -2363382.333333
## Step 6: adding 21701388.888889, exp(-50.000000) = 19338006.555556
## Step 7: adding -155009920.634921, exp(-50.000000) = -135671914.079365
## Step 8: adding 968812003.968254, exp(-50.000000) = 833140089.888889
## Step 9: adding -5382288910.934744, exp(-50.000000) = -4549148821.045855
## Step 10: adding 26911444554.673717, exp(-50.000000) = 22362295733.627861
## Step 11: adding -122324747975.789612, exp(-50.000000) = -99962452242.161743
## Step 12: adding 509686449899.123352, exp(-50.000000) = 409723997656.961609
## Step 13: adding -1960332499612.012939, exp(-50.000000) = -1550608501955.051270
## Step 14: adding 7001187498614.331055, exp(-50.000000) = 5450578996659.279297
## Step 15: adding -23337291662047.769531, exp(-50.000000) = -17886712665388.492188
## Step 16: adding 72929036443899.281250, exp(-50.000000) = 55042323778510.789062
## Step 17: adding -214497166011468.468750, exp(-50.000000) = -159454842232957.687500
```

```
## Step 18: adding 595825461142968.000000, exp(-50.000000) = 436370618910010.312500
## Step 19: adding -1567961739849915.750000, exp(-50.000000) = -1131591120939905.500000
## Step 20: adding 3919904349624789.500000, exp(-50.000000) = 2788313228684884.000000
## Step 21: adding -9333105594344738.000000, exp(-50.000000) = -6544792365659854.000000
## Step 22: adding 21211603623510768.000000, exp(-50.000000) = 14666811257850914.000000
## Step 23: adding -46112181790240800.000000, exp(-50.000000) = -31445370532389888.000000
## Step 24: adding 96067045396334992.000000, exp(-50.000000) = 64621674863945104.000000
## Step 25: adding -192134090792670016.000000, exp(-50.000000) = -127512415928724912.000000
## Step 26: adding 369488636139750016.000000, exp(-50.000000) = 241976220211025088.000000
## Step 27: adding -684238215073611136.000000, exp(-50.000000) = -442261994862586048.000000
## Step 28: adding 1221853955488591360.000000, exp(-50.000000) = 779591960626005248.000000
## Step 29: adding -2106644750842398976.000000, exp(-50.000000) = -1327052790216393728.000000
## Step 30: adding 3511074584737331712.000000, exp(-50.000000) = 2184021794520937984.000000
## Step 31: adding -5663023523769889792.000000, exp(-50.000000) = -3479001729248951808.000000
## Step 32: adding 8848474255890452480.000000, exp(-50.000000) = 5369472526641500160.000000
## Step 33: adding -13406779175591593984.000000, exp(-50.000000) = -8037306648950093824.000000
## Step 34: adding 19715851728811167744.000000, exp(-50.000000) = 11678545079861073920.000000
## Step 35: adding -28165502469730238464.000000, exp(-50.000000) = -16486957389869164544.000000
## Step 36: adding 39118753430180888576.000000, exp(-50.000000) = 22631796040311726080.000000
## Step 37: adding -52863180311055253504.000000, exp(-50.000000) = -30231384270743527424.000000
## Step 38: adding 69556816198756909056.000000, exp(-50.000000) = 39325431928013381632.000000
## Step 39: adding -89175405383021674496.000000, exp(-50.000000) = -49849973455008292864.000000
## Step 40: adding 111469256728777097216.000000, exp(-50.000000) = 61619283273768804352.000000
## Step 41: adding -135938117961923280896.000000, exp(-50.000000) = -74318834688154468352.000000
## Step 42: adding 161831092811813421056.000000, exp(-50.000000) = 87512258123658952704.000000
## Step 43: adding -188175689316062101504.000000, exp(-50.000000) = -100663431192403148800.000000
## Step 44: adding 213836010586434207744.000000, exp(-50.000000) = 113172579394031058944.000000
## Step 45: adding -237595567318260219904.000000, exp(-50.000000) = -124422987924229160960.000000
## Step 46: adding 258256051432891547648.000000, exp(-50.000000) = 133833063508662386688.000000
## Step 47: adding -274740480247756980224.000000, exp(-50.000000) = -140907416739094593536.000000
## Step 48: adding 286188000258080210944.000000, exp(-50.000000) = 145280583518985617408.000000
## Step 49: adding -292028571691918589952.000000, exp(-50.000000) = -146747988172932972544.000000
## Step 50: adding 29202857169191858952.000000, exp(-50.000000) = 145280583518985617408.000000
## Step 51: adding -286302521266586877952.000000, exp(-50.000000) = -141021937747601260544.000000
## Step 52: adding 275290885833256599552.000000, exp(-50.000000) = 134268948085655339008.000000
## Step 53: adding -259708382861562839040.000000, exp(-50.000000) = -125439434775907500032.000000
## Step 54: adding 24047072487181746176.000000, exp(-50.000000) = 11503129009590961728.000000
## Step 55: adding -21860974988347041382.4.000000, exp(-50.000000) = -10357845978756045209 6.000000
## Step 56: adding 195187276681670000640.000000, exp(-50.000000) = 91608816894109548544.000000
## Step 57: adding -171216909369885949952.000000, exp(-50.000000) = -79608092475776401408.000000
## Step 58: adding 147600783939556835328.000000, exp(-50.000000) = 67992691463780433920.000000
## Step 59: adding -125085410118268502016.000000, exp(-50.000000) = -57092718654488068096.000000
## Step 60: adding 104237841765223743488.000000, exp(-50.000000) = 47145123110735675392.000000
## Step 61: adding -85440853905921097728.000000, exp(-50.000000) = -38295730795185422336.000000
## Step 62: adding 68903914440258953216.000000, exp(-50.000000) = 30608183645073530880.000000
## Step 63: adding -54685646381157900288.000000, exp(-50.000000) = -24077462736084369408.000000
## Step 64: adding 42723161235279609856.000000, exp(-50.000000) = 18645698499195240448.000000
## Step 65: adding -32863970180984315904.000000, exp(-50.000000) = -14218271681789075456.000000
## Step 66: adding 24896947106806300672.000000, exp(-50.000000) = 10678675425017225216.000000
## Step 67: adding -18579811273736044544.000000, exp(-50.000000) = -7901135848718819328.000000
## Step 68: adding 13661625936570621952.000000, exp(-50.000000) = 5760490087851802624.000000
## Step 69: adding -9899728939543928832.000000, exp(-50.000000) = -4139238851692126208.000000
## Step 70: adding 7071234956817091584.000000, exp(-50.000000) = 2931996105124965376.000000
## Step 71: adding -4979742927335980032.000000, exp(-50.000000) = -2047746822211014656.000000
```

```
## Step 72: adding 3458154810649986048.000000, exp(-50.000000) = 1410407988438971392.000000
## Step 73: adding -2368599185376702976.000000, exp(-50.000000) = -958191196937731584.000000
## Step 74: adding 1600404854984258816.000000, exp(-50.000000) = 642213658046527232.000000
## Step 75: adding -1066936569989505792.000000, exp(-50.000000) = -424722911942978560.000000
## Step 76: adding 701931953940464384.000000, exp(-50.000000) = 277209041997485824.000000
## Step 77: adding -455799970091210624.000000, exp(-50.000000) = -178590928093724800.000000
## Step 78: adding 292179468007186304.000000, exp(-50.000000) = 113588539913461504.000000
## Step 79: adding -184923713928598944.000000, exp(-50.000000) = -71335174015137440.000000
## Step 80: adding 115577321205374336.000000, exp(-50.000000) = 44242147190236896.000000
## Step 81: adding -71344025435416256.000000, exp(-50.000000) = -27101878245179360.000000
## Step 82: adding 43502454533790400.000000, exp(-50.000000) = 16400576288611040.000000
## Step 83: adding -26206297911921928.000000, exp(-50.000000) = -9805721623310888.000000
## Step 84: adding 15598986852334482.000000, exp(-50.000000) = 5793265229023594.000000
## Step 85: adding -9175874619020284.000000, exp(-50.000000) = -3382609389996690.000000
## Step 86: adding 5334810825011793.000000, exp(-50.000000) = 1952201435015103.000000
## Step 87: adding -3065983232765398.500000, exp(-50.000000) = -1113781797750295.500000
## Step 88: adding 1742035927707612.750000, exp(-50.000000) = 628254129957317.250000
## Step 89: adding -978671869498658.875000, exp(-50.000000) = -350417739541341.625000
## Step 90: adding 543706594165921.625000, exp(-50.000000) = 193288854624580.000000
## Step 91: adding -298739886904352.500000, exp(-50.000000) = -105451032279772.500000
## Step 92: adding 162358634187148.093750, exp(-50.000000) = 56907601907375.593750
## Step 93: adding -87289588272660.265625, exp(-50.000000) = -30381986365284.671875
## Step 94: adding 46430632059925.671875, exp(-50.000000) = 16048645694641.000000
## Step 95: adding -24437174768381.929688, exp(-50.000000) = -8388529073740.929688
## Step 96: adding 12727695191865.587891, exp(-50.000000) = 4339166118124.658203
## Step 97: adding -6560667624672.983398, exp(-50.000000) = -2221501506548.325195
## Step 98: adding 3347279400343.358887, exp(-50.000000) = 1125777893795.033691
## Step 99: adding -1690545151688.564941, exp(-50.000000) = -564767257893.531250
## Step 100: adding 845272575844.282471, exp(-50.000000) = 280505317950.751221
## Step 101: adding -418451770219.941833, exp(-50.000000) = -137946452269.190613
## Step 102: adding 205123416774.481262, exp(-50.000000) = 67176964505.290649
## Step 103: adding -99574474162.369537, exp(-50.000000) = -32397509657.078888
## Step 104: adding 47872343347.293045, exp(-50.000000) = 15474833690.214157
## Step 105: adding -22796353974.901451, exp(-50.000000) = -7321520284.687294
## Step 106: adding 10752997157.972382, exp(-50.000000) = 3431476873.285088
## Step 107: adding -5024765027.089898, exp(-50.000000) = -1593288153.804811
## Step 108: adding 2326280105.134212, exp(-50.000000) = 732991951.329401
## Step 109: adding -1067100965.657895, exp(-50.000000) = -334109014.328494
## Step 110: adding 485045893.480862, exp(-50.000000) = 150936879.152368
## Step 111: adding -218489141.207595, exp(-50.000000) = -67552262.055228
## Step 112: adding 97539795.181962, exp(-50.000000) = 29987533.126735
## Step 113: adding -43159201.407948, exp(-50.000000) = -13171668.281213
## Step 114: adding 18929474.301732, exp(-50.000000) = 5757806.020518
## Step 115: adding -8230206.218144, exp(-50.000000) = -2472400.197626
## Step 116: adding 3547502.680235, exp(-50.000000) = 1075102.482609
## Step 117: adding -1516026.786425, exp(-50.000000) = -440924.303816
## Step 118: adding 642384.231536, exp(-50.000000) = 201459.927720
## Step 119: adding -269909.340982, exp(-50.000000) = -68449.413262
## Step 120: adding 112462.225409, exp(-50.000000) = 44012.812147
## Step 121: adding -46471.993971, exp(-50.000000) = -2459.181824
## Step 122: adding 19045.899168, exp(-50.000000) = 16586.717345
## Step 123: adding -7742.235434, exp(-50.000000) = 8844.481910
## Step 124: adding 3121.869127, exp(-50.000000) = 11966.351037
## Step 125: adding -1248.747651, exp(-50.000000) = 10717.603386
```

```
## Step 126: adding 495.534782, exp(-50.000000) = 11213.138168
## Step 127: adding -195.092434, exp(-50.000000) = 11018.045735
## Step 128: adding 76.207982, exp(-50.000000) = 11094.253717
## Step 129: adding -29.537978, exp(-50.000000) = 11064.715739
## Step 130: adding 11.360761, exp(-50.000000) = 11076.076500
## Step 131: adding -4.336168, exp(-50.000000) = 11071.740331
## Step 132: adding 1.642488, exp(-50.000000) = 11073.382819
## Step 133: adding -0.617477, exp(-50.000000) = 11072.765343
## Step 134: adding 0.230402, exp(-50.000000) = 11072.995745
## Step 135: adding -0.085334, exp(-50.000000) = 11072.910411
## Step 136: adding 0.031373, exp(-50.000000) = 11072.941783
## Step 137: adding -0.011450, exp(-50.000000) = 11072.930333
## Step 138: adding 0.004149, exp(-50.000000) = 11072.934482
## Step 139: adding -0.001492, exp(-50.000000) = 11072.932990
## Step 140: adding 0.000533, exp(-50.000000) = 11072.933523
## Step 141: adding -0.000189, exp(-50.000000) = 11072.933334
## Step 142: adding 0.000067, exp(-50.000000) = 11072.933400
## Step 143: adding -0.000023, exp(-50.000000) = 11072.933377
## Step 144: adding 0.000008, exp(-50.000000) = 11072.933385
## Step 145: adding -0.000003, exp(-50.000000) = 11072.933382
## Step 146: adding 0.000001, exp(-50.000000) = 11072.933383
## Step 147: adding -0.000000, exp(-50.000000) = 11072.933383
## Step 148: adding 0.000000, exp(-50.000000) = 11072.933383
## Step 149: adding -0.000000, exp(-50.000000) = 11072.933383
## Step 150: adding 0.000000, exp(-50.000000) = 11072.933383
## Step 151: adding -0.000000, exp(-50.000000) = 11072.933383
## Step 152: adding 0.000000, exp(-50.000000) = 11072.933383
## Step 153: adding -0.000000, exp(-50.000000) = 11072.933383
## Step 154: adding 0.000000, exp(-50.000000) = 11072.933383
## Step 155: adding -0.000000, exp(-50.000000) = 11072.933383
## Step 156: adding 0.000000, exp(-50.000000) = 11072.933383
## Step 157: adding -0.000000, exp(-50.000000) = 11072.933383
## Step 158: adding 0.000000, exp(-50.000000) = 11072.933383
## Step 159: adding -0.000000, exp(-50.000000) = 11072.933383
## Step 160: adding 0.000000, exp(-50.000000) = 11072.933383
## Step 161: adding -0.000000, exp(-50.000000) = 11072.933383
## Step 162: adding 0.000000, exp(-50.000000) = 11072.933383
## Step 163: adding -0.000000, exp(-50.000000) = 11072.933383
## Step 164: adding 0.000000, exp(-50.000000) = 11072.933383
## Step 165: adding -0.000000, exp(-50.000000) = 11072.933383
## Step 166: adding 0.000000, exp(-50.000000) = 11072.933383
## Step 167: adding -0.000000, exp(-50.000000) = 11072.933383

## [1] 11072.9333828919698135
```

```
## using another expression to avoid computing large - large
fexp2 <- function(x) {
    xa <- abs(x)
    i <- 0
    expx <- 1
    u <- 1
    while(u>1e-20*expx) {
        i <- i+1
        u <- u*xa/i
        expx <- expx+u
```

```
    }
    if (x >= 0) expx else 1/expx
}
```

```
matrix(c(exp(-10),fexp2(-10)))
```

```
##                               [,1]
## [1,] 4.539992976248485417315e-05
## [2,] 4.539992976248486094941e-05
```

```
matrix(c(exp(-20),fexp2(-20)))
```

```
##                               [,1]
## [1,] 2.061153622438557869942e-09
## [2,] 2.061153622438558283532e-09
```

```
matrix(c(exp(-50),fexp2(-50)))
```

```
##                               [,1]
## [1,] 1.928749847963917820563e-22
## [2,] 1.928749847963916880168e-22
```

3.3 Computing variance: avoiding computing large - large

```
var1 <- function(x)
{ n <- length(x)
( sum(x^2) - sum(x)^2/n ) / (n-1)
}

var2 <- function(x)
{
    n <- length(x)

    sum((x-mean(x))^2) / (n-1)
}

x <- c(1,2,3)

c(var1(x),var2(x),var(x))
```

```
## [1] 1 1 1
```

```
x <- c(1,2,3) + 1e10

c(var1(x),var2(x),var(x))
```

```
## [1] -32768      1      1
```

4 Overflow and underflow

```
## demo of overflow problem: avoid Inf/Inf, 0/0
p1 <- function(theta)
{
    exp(theta)/(1+exp(theta))
}

p2 <- function(theta)
{
    1/(1+exp(-theta))
}


theta <- 2000

p1(theta)
```

```
## [1] NaN
```

```
p2(theta)
```

```
## [1] 1
```

```
theta <- -2000

p1(theta)
```

```
## [1] 0
```

```
p2(theta)
```

```
## [1] 0
```

5 Use logarithm to handle overflow and underflow numbers

```
log_x <- 800

x <- exp(log_x)

x
```

```
## [1] Inf
```

```
log_y <- 805

y <- exp(log_y)

y
```

```
## [1] Inf
```

```
x/y
```

```
## [1] NaN
```

```r
log_xovery <- log_x- log_y

log_xovery
```

```
## [1] -5
```

```r
exp(log_xovery)
```

```
## [1] 0.006737946999085467000845
```

```r
#looking for log of sum of numbers in logarithm

log_sum_exp <- function(log_x)
{
    max_log_x <- max(log_x)

    max_log_x + log( sum(exp(log_x - max_log_x) ))
}

log_x <- c(2000,2010,2030)

exp(log_x)
```

```
## [1] Inf Inf Inf
```

```r
log_sum_exp(log_x)
```

```
## [1] 2030.000000002061142368
```

```r
log_x <- - c(2000,2010,2030)

exp(log_x)
```

```
## [1] 0 0 0
```

```r
log_sum_exp(log_x)
```

```
## [1] -1999.99995460110062595
```

```r
log_minus_exp <- function(log_x,log_y)

{  if(log_x < log_y)
    stop("The first argument is bigger than the second")

    log_x + log(1-exp(log_y-log_x))

}

log_minus_exp(2020,2000)
```

```
## [1] 2019.999999997938857632
```

```r
exp(2000) - exp(1999)
```

```
## [1] NaN
```