Stærðfræði og reiknifræði – Skilaverkefni 5

Leysa skal tvö dæmi af þremur til að fá fullt hús stiga (valin tvö bestu ef þið leysið öll). Skilið næsta mánudag, 17. febrúar (í allra síðasta lagi fyrir kl. 10 um kvöldið)

In [1]:
```
#BYRJA -- Keyrið til að frumstilla.
import numpy as np
import numpy.random as npr
import scipy.stats as stat
import matplotlib.pyplot as plt
plt.rc('axes', axisbelow=True)
%matplotlib inline
# disp(x,y...) skrifar x,y... með 3 aukastöfum
def disp(*args): print(*(f'{a:.5f}' if isinstance(a,float) else a for a in args))
np.set_printoptions(precision=3, floatmode='fixed', suppress=True)
```

In [2]:
```
%%javascript
MathJax.Hub.Config({TeX: {equationNumbers:{autoNumber:"none"}, TagSide:"left"}});
```

S5-A Þéttifall normaldreifingar

Þetta dæmi sameinar að vera smá Python æfing, að gefa innsýn í tölfræði, og að gefa okkur fall til að nota í dæmi S5-B.

Þéttifall normaldreifingar, f, er þannig að líkurnar á að normaldreifð slembistærð sé á tilteknu bili eru heildið af fallinu (= flatarmálið undir ferlinum) yfir bilið:
$$ P(a \leq X \leq b) = \int_{a}^{b} f(x) \, dx\tag{1} $$

Heildarflatarmálið undir ferlinum er 1, $P(X \leq a) = \int_{-\infty}^{a} f(x) \, dx$ og $P(X \geq a) = \int_{a}^{\infty} f(x) \, dx$

Á ensku er stundum talað um *bell curve*, og á Math is fun (https://www.mathsisfun.com/data/standard-normal-distribution.html) er ágæt umfjöllun. Þegar meðaltalið er $\mu = 0$ og staðalfrávikið er $\sigma = 1$ er fallið
$$ f(x) = \frac{1}{\sqrt{2\pi}} \exp(-\tfrac{1}{2}x^2)\tag{2} $$

Eins og útskýrt er á *Math is fun* er hægt að nota sama fall fyrir almenn μ og σ ef við hugsum okkur að x-ás mæli hve mörgum staðalfrávikum maður er fyrir ofan eða neðan meðaltalið.

Tökum dæmi af amerísku körfuboltamönnunum. Meðalhæð þeirra er 187 cm og staðalfrávikið 7 cm. Ef hæðin er normaldreifð (sem er líklegt) þá eru 95% líkur á að körfuboltamaður sé milli 173 og 201 cm (sem sé ± 2 staðalfrávik) skv. "Math is fun". Með öðrum orðum gildir:
$$\tag{3} \int_{-2}^{2} f(x) \, dx \approx 0.95 $$

Í Python er hægt að reikna þessar líkur eins og sýnt er í eftirfarandi reit, sem líka reiknar líkur á að körfuboltamaður sé > 220 cm (nánast engar) og < 160 cm (sáralitlar).

In [3]:
```
#SCIPY-NORMAL Normaldreifing í scipy.stats
nd = stat.norm(loc=187, scale=7)
p1 = nd.cdf(201) - nd.cdf(173)
p2 = 1 - nd.cdf(220)
p3 = nd.cdf(168)
disp('P(milli 173 og 201) =', p1*100, '%')
disp('P(stærri en 220) =', p2*100, '%')
disp('P(minni en 168) =', p3*100, '%')
nstd = stat.norm()
nstd.pdf(0)
```

```
P(milli 173 og 201) = 95.44997 %
P(stærri en 220) = 0.00012 %
P(minni en 168) = 0.33209 %
```

Out[3]:
```
0.3989422804014327
```

Þá er það verkefnið:

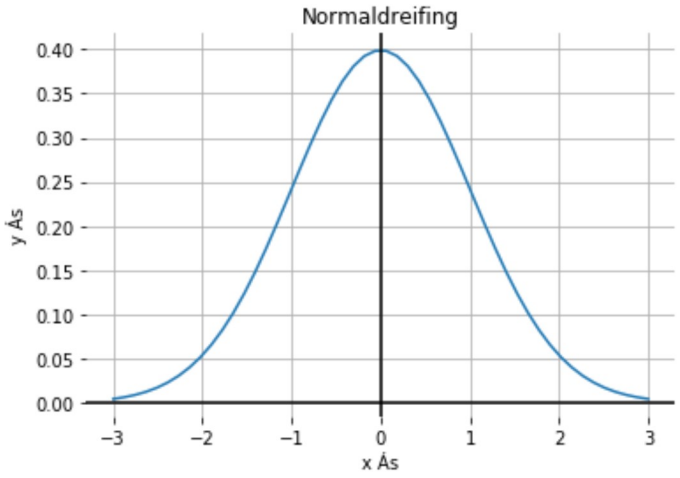
1. Skrifið fallið (2) í Python. Reiknið $f(0)$ (sem ætti að gefa 0.3989) og $f(3)$.
2. Búið til vandað graf af f á bilinu $[-3,3]$ (sjá grein 2.3.6 (https://cs.hi.is/strei/kafli02.html#teikning-af-grofum-falla) í fyrirlestrarnótum – teiknið svarta x- og y-ása, merkið með `xlabel` og `ylabel`, setjið titil á myndina og rúðunet)

In [35]:
```
#A1
def f(x):
    x = (1/np.sqrt(2*np.pi))*(np.exp(-(1/2)*x**2))
    return x

null = 0
thrir = 3

print(f(0))
print(f(3))
```

```
0.3989422804014327
0.004431848411938007 5
```

In [36]:
```
#A2
t = np.linspace(-3, 3)
plt.plot(t, f(t))
plt.grid()
plt.axvline(c='k')
plt.axhline(c='k')
plt.box(False)
plt.xlabel('x Ás')
plt.ylabel('y Ás')
plt.title('Normaldreifing')
```

Out[36]:
```
Text(0.5, 1.0, 'Normaldreifing')
```



S5-B Simpsons-regla

In [39]:
```
#TRAP
def trap(f,a,b,n=10):
    """Nálgar heildið af f frá a til b með trapisureglu og n hlutbilum"""
    dx = (b-a)/n
    c = 2*np.ones(n+1)
    c[0] = c[n] = 1
    x = np.linspace(a,b,n+1)
    F = f(x)
    H = (c @ F) * dx/2
    return H

T = trap(f,-2,2,246)
disp(T)
```

```
0.95449
```

Í tímadæmum 5 var trapisuregla til að reikna heildi á dagskrá og útfærsla úr lausn þess er í reitnum hér að ofan. Hér kynnumst við annarri reglu, sem kölluð er *Simpsons-regla*. Í trapisureglu er heildisbilinu skipt í n hlutbil, fallið sem heilda skal nálgað með beinum línustrikum og heildi þess nálgað með flatarmálinu undir þessum línustrikum. Í Simpsonsreglu er fallið hinsvegar nálgað (eða *brúað* eins og það er kallað) með parabólum og heildið nálgað með flatarmálinu undir þeim. Áhugasamir geta lesið meira t.d. í Wikipediu grein um aðferðina (https://en.wikipedia.org/wiki/Simpson%27s_rule).

Simpsons-formúlan er eftirfarandi:

$$\notag \int_{a}^{b} f(x) \, dx \approx \frac{\Delta x}{3}\left(f(x_0) + 4f(x_1)+2f(x_2)+ 4f(x_3)+2f(x_4)+\cdots+4f(x_{n-1}) + f(x_{n})\right) $$

þar sem $\Delta x = \displaystyle{\frac{b-a}{n}}$ og $x_i=a+i\Delta x$, $i = 0,\cdots n$.

Verkefnið:

1. Skrifið fall `simpson(f,a,b,n)` sem nálgar heildi með Simpsons-reglu.

In [74]:
```
#B1
def simpson(f, a, b, n):
    sum = f(a) + f(b)
    dx = (b-a)/n

    for i in range(1,n):
        xi = a + i*dx
        if i%2 == 0:
            sum += 2*f(xi)
        else:
            sum += 4*f(xi)
    return sum*dx/3
```

Fallið f sem gefið er með (2) er ekki heildanlegt með venjulegum stærðfræðigreiningaraðferðum heldur þarf að nálga heildi eins og (1), til dæmis með Simpsons-reglu.

1. Ákvarðið hvaða n þarf að nota með `simpson` til að ná sömu nákvæmni í reikningi heildisins (3) og sýnd er í reit `#SCIPY-NORMAL`, sem sé 0.95449.

In [97]:
```
#B2
print(simpson(f,-2,2,14))

# Svar: n=14
```

```
0.9544910124029714
```

1. Svarið b-lið miðað við að Trapisuregla sé notuð.

In [114]:
```
#B3
print(trap(f,-2,2,172))

#Svar: n=172
```

```
0.9544898886567318
```

S5-C Lágmörkun með stigli

1. Í skiladæmum 4 var fall Rosenbrocks lágmarkað, og í fyrirlestraræfingu var stigull (*gradient*) þess reiknaður. Fallið er:

   $$ f(x, y) = (1 - x)^2 + 100(y - x^2)^2 $$

   og stigull þess er:

   $$ \nabla f(x,y) = \pmatrix{2x - 2 - 400x(y - x^2) \\ 200(y - x^2)} $$

   Ef við umritum formúlurnar með því að skrifa (x_0, y_0) í stað (x,y) og skilgreina $x = (x_0, y_0)$ fæst

   $$ \begin{align}f(x) &= (1-x_0)^2 + 100(x_1 - x_0^2)^2 \\ \nabla f(x) &= \pmatrix{2x_0 - 2 - 400x_0(x_1 - x_0^2) \\ 200(x_1 - x_0^2)}\end{align} $$

   Skrifið Python föll `f(x)` og `g(x)` sem reikna fallsgildi og stigul Rosenbrock-fallsins.

In [ ]:
```
#C1
```

1. `opt.minimize` hefur valkvæðan stika `jac` sem er nafn falls sem reiknar stigul fallsins sem á að lágmarka (*jac* er stytting á *Jacobian*, sem fyrir $f\colon\Bbb{R}\to\Bbb{R^2}$ er samheiti við *gradient*). Með *jac*-stika verður lágmörkunarkallið:

   ```
   result = opt.minimize(f, x0, jac=g)
   ```

   Ef `opt.minimize` hefur aðgang að stiglinum þá þarf mun færri köll á fallið til að finna lággildið. Ákvarðið lággildi Rosenbrock-fallsins ef byrjað er í $x_0=(-1.2, 1)$ bæði með og án stigul-falls, og finnið út hve margar ítrekanir og köll á `f` þarf í hvoru tilviki fyrir sig.

In [9]:
```
#C2
```