Stærðfræði og reiknifræði – Skilaverkefni 4

Leysið verkefnin með því að færa lausnir hér inn í þessa Júpíter-bók, búa til úr henni PDF-skjal (Hægri-smella og Print-to-PDF í vafra kemur best út) og hlaða því inn í Canvas. Skilafrestur er á mánudag 10. feb. kl. 22:00, eftir það lokast á skil.

Í þessu verkefni megið þið gjarnan hjálpast að, en eins og fyrr þarf hver fyrir sig að skila sinni lausn. Getið vinnufélaga í svari við "Hvernig gekk spurningu aftast" (ekki sleppa henni). Það er bannað að fá lánaðar tilbúnar lausnir eða lána öðrum.

In [78]:

```
#BYRJA -- Keyrið til að frumstilla numpy o.fl.
import numpy as np, numpy.random as npr
import matplotlib.pyplot as plt, scipy.stats as st
from scipy.optimize import minimize
plt.rc('axes', axisbelow=True)
%matplotlib inline
np.set_printoptions(precision=4, floatmode='fixed', suppress=True)

def plotline(a,b):
    """Teiknar línu y = ax + b inn á mynd sem er virk"""
    X = np.array(plt.xlim())
    Y = a*X + b
    plt.plot(X,Y,lw=3)

def plotpara(a,b,c):
    """Teiknar parabólu y = ax^2 + bx + c inn á mynd sem er virk"""
    x = np.linspace(*plt.xlim())
    y = a*x**2 + b*x + c
    plt.plot(x,y,lw=3)
```

S4.1 Lágmörkun tvívíðs falls

Í tímadæmum 4 kynntumst við lágmörkun á tvívíðu falli með `minimize` úr pakkanum `scipy.optimize`. Einföld notkun, sem finnur og prentar út lággildispunktinn er:

```
result = minimize(f,x0)
print(result)
```

þar sem `f` er fallið sem er lágmarkað og `x0` er upphafságiskun á lausn.

Fall Rosenbrocks kemur við sögu í fyrirlestrarnótunum, í kafla 3.3.6. Það er oft notað sem dæmi í lágmörkun falla. Skilgreining þess er:
$$ f(x_0, x_1) = (1 - x_0)^2 + 100(x_1 - x_0^2)^2 $$

Teikning hæðarlína var líka á dagskrá í tímadæmi T4.4 og í grein 3.2.3 (https://cs.hi.is/strei/kafli03.html#haearlinur-me-skipuninni-contour) í fyrirlestrarnótum.

Verkefnið:

1. Skrifið fall, `ros(x)` sem reiknar fall Rosenbrocks í Python.
2. Teiknið hæðarlínur fallsins `ros` á rétthyrningnum $[-1.5, 1.5] \times [-0.5, 2.5]$. Til að fá sæmilega mynd þarf nota svolítið marga punkta í `np.linspace` (t.d. 120) og auk þess að láta `plt.contour` fá handvalinn `level` -stika. Þessi dugar t.d.:

   ```
   L = np.linspace(0,26,14)**2
   L[0] = 0.2
   ```

   Teiknið svo lággildispunktinn $(1,1)$ inn á myndina, grænan og sæmilega stóran.
3. Notið `minimize` til að ákvarða lággildið ef byrjað er með `x0` = [-1.2,2]. `minimize` getur tekið inn viðbótarstika `callback=cb`, og þá kallar það á `cb(xk)` með núverandi ítrekunargildi `xk` í hverri ítrekun (í þessu tilviki er `xk` tveggja staka vigur). Notið þennan "fídus" til að teikna ítrekunargildin (t.d. með `plt.scatter(x[0],x[1])` inni í `cb`).
4. `minimize` getur líka tekið inn stika `method="aðferð"`, þar sem aðferð getur t.d. verið 'L-BFGS-B', 'CG', og 'Powell' fyrir utan þá sjálfvöldu, 'BFGS'. Prófið og segið frá niðurstöðum (ath. að `result.nit` gefur fjölda ítrekana og `result.nfev` sem gefur fjölda kalla á `ros`).

In [155]:
```
## 1 - ros(x) - reiknar Rosenbrocks
def ros(x):
    w = x[:,0]
    y = x[:,1]
    return((1-w)**2+100(y-w**2)**2)

## 2 - Teikna hæðarlínur
u=np.linspace(-1.5,1.5)
v=np.linspace(-0.5,2.5)
Z=np.array([ros(x) for x in u])
Y=np.array([ros(x) for x in v])
plt.contour(u, v, Z.T, 20)


## 3 - Minimize
x0 = np.array([-1.2,1])
result = minimize(s,x0)

## 4 -
```

```
---------------------------------------------------------------------------
IndexError                                Traceback (most recent call last)
<ipython-input-155-0bda1b4bf475> in <module>
      8 u=np.linspace(-1.5,1.5)
      9 v=np.linspace(-0.5,2.5)
---> 10 Z=np.array([ros(x) for x in u])
     11 Y=np.array([ros(x) for x in v])
     12 plt.contour(u, v, Z.T, 20)

<ipython-input-155-0bda1b4bf475> in <listcomp>(.0)
      8 u=np.linspace(-1.5,1.5)
      9 v=np.linspace(-0.5,2.5)
---> 10 Z=np.array([ros(x) for x in u])
     11 Y=np.array([ros(x) for x in v])
     12 plt.contour(u, v, Z.T, 20)

<ipython-input-155-0bda1b4bf475> in ros(x)
      1 ## 1 - ros(x) - reiknar Rosenbrocks
      2 def ros(x):
----> 3     w = x[:,0]
      4     y = x[:,1]
      5     return((1-w)**2+100(y-w**2)**2)

IndexError: invalid index to scalar variable.
```

S4.2 Hiti og úrkoma

Innlestur gagna: Skráin [cs.hi.is/strei/hiti-urkoma.txt (http://cs.hi.is/strei/hiti-urkoma.txt)](http://cs.hi.is/strei/hiti-urkoma.txt) geymir ársmeðalhita og heildarúrkomu áranna 1949–2018 í Stykkishólmi og byrjar svona:

```
1949    3.2 565.5
1950    4.0 535.5
1951    3.4 460.6
...
```

Með hliðsjón af Tímadæmum 3.5 má lesa gögnin inn í Python með:

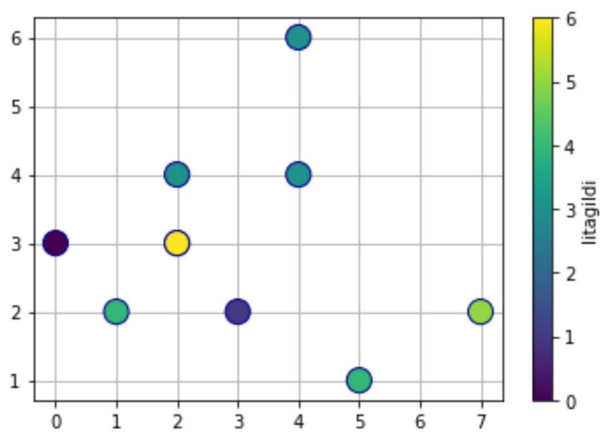
```
f = urlopen('http://cs.hi.is/strei/hiti-urkoma.txt')
(ár,h,r) = np.loadtxt(f).T
```

Skatterplott með litakóða: í greinum [2.3.2 (https://cs.hi.is/strei/kafli02.html#einfaldar-myndir)](https://cs.hi.is/strei/kafli02.html#einfaldar-myndir) og [2.3.7 (https://cs.hi.is/strei/kafli02.html#teikning-punktasafns-og-jafna-bestu-linu)](https://cs.hi.is/strei/kafli02.html#teikning-punktasafns-og-jafna-bestu-linu) er fjallað um *skatterplott*-skipunina sem kalla má á með:

```
plt.scatter(x, y, c=litur, s=stærð, edgecolor=randlitur)
```

Þar var hinsvegar ekki nefnt að `c` má vera vigur (og raunar `s` líka), og svo má kalla á `plt.colorbar()` til að sjá til hvaða talnagilda litirnir svara. Hér er dæmi sem notar alla stikana:

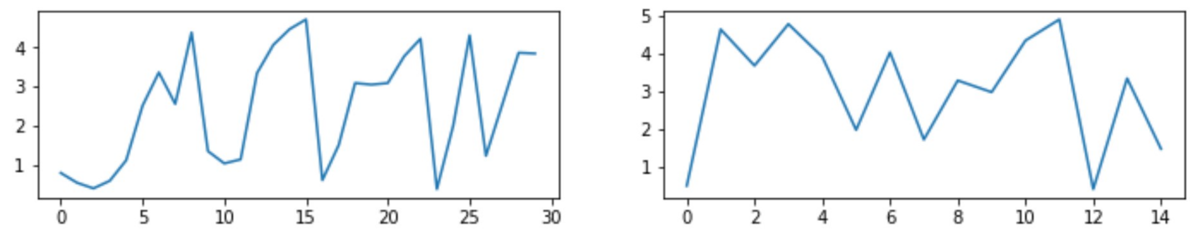
In [13]:
```
x = np.array([5,7,2,3,0,2,4,1,2,4])
y = np.array([1,2,3,2,3,4,6,2,3,4])
z = np.array([4,5,2,1,0,3,3,4,6,3])
plt.scatter(x, y, c=z, s=200, edgecolor='darkblue')
plt.colorbar(label='litagildi'); plt.grid()
```



(svo er hægt að skipta litaskalanum (*colormap*) út, en bíðum með að tala um það).

Myndir hlið við hlið: Annað trix sem ekki hefur verið rætt er að teikna tvö (eða fleiri) línurit hlið við hlið í samsettri mynd. Þá er notað `plt.subplot(1,n,i)` sem velur `i` -ta línuritið af `n` (byrjar að telja í 1 á ópæþónskan máta). Hér er einfalt dæmi:

In [17]:
```
plt.figure(figsize = (12,2))
plt.subplot(1,2,1); plt.plot(5*npr.random(30))
plt.subplot(1,2,2); plt.plot(5*npr.random(15));
```



Um fylgni: Í byrjunarreitnum er aðaltölfræðipakkinn fyrir Python fluttur inn með `import scipy.stats as st`. Eitt af því sem hann ræður við er að reikna fylgnistuðul tveggja vigra, `x` og `y`, og reyndar í leiðinni p-gildi fyrir þá tilgátu að fylgnistuðullinn sé ekki 0:

```
(r,p) = st.pearsonr(x, y)
```

Ef $p = 0.05$ þá er nokkuð ólíklegt að fylgnin sé í raun engin og og það sé bara fyrir tilviljun að sem við fáum jákvæðan eða neikvæðan fylgnistuðul: Það mundi bara gerast í tuttugasta hvert skipti að við fengjum meiri fylgni úr gögnum sem eru í raun óháð. Og ef p er minna, t.d. 0.005 þá er það orðið mjög ólíklegt. Oft er talað um að fylgni sé marktæk ef p er lítið, og p er þá kallað marktæknistig. Stundum er rúnnuð tala sem er stærri en p getið innan sviga: *Fylgnin er marktæk (p < 0.01)*. Hæg er að lesa um fylgni hér (https://www.spss-tutorials.com/pearson-correlation-coefficient/) eða á Wikipediu (https://en.wikipedia.org/wiki/Pearson_correlation_coefficient) (dálítið flókin grein) og líka æft ykkur að meta fylgni (https://en.wikiversity.org/wiki/Survey_research_and_design_in_psychology/Tutorials/Correlation/Scatterplot_correlation_guess).

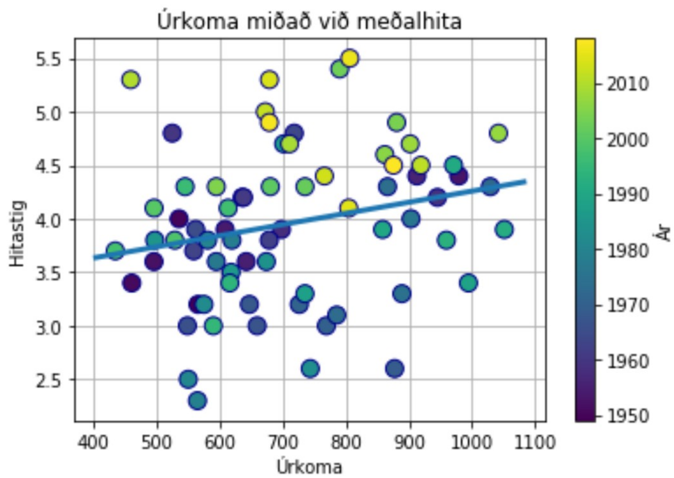
Verkefnið

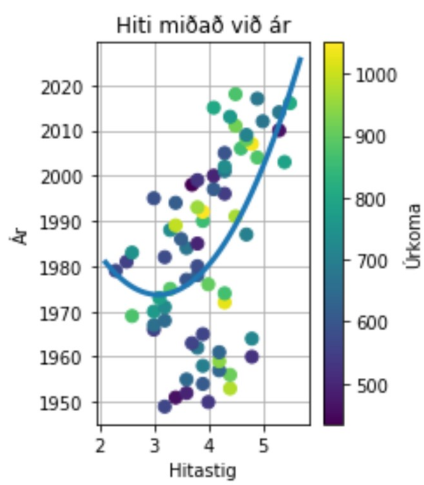
1. Teiknið *skatterplott* af úrkomu (á y-ás) á móti hita í vinstri hlutmynd (*subplot*). Látið ártalið stjórna lit. Setjið inn merkingar á ása, rúðunet, titil, *colorbar*. Veljið hæfilega punktastærð. Reiknið líka jöfnu bestu línu (sbr. kafla 2.3.6 (https://cs.hi.is/strei/kafli02.html#teikning-punktasafns-og-jafna-bestu-linu) í fyrirlestrarnótum) og teiknið hana inn á myndina með fallinu `plotline` sem skilgreint er í reitnum #BYRJA að ofan.
2. Finnið fylgni ársúrkomu og ársmeðalhita í Stykkishólmi og p-gildi hennar. Er fylgnin marktæk? (þið megið gjarnan búa til textareit og svara í honum).
3. Teiknið *skatterplott* af meðalhita á móti ári í hægri hlutmynd (*subplot*) og látið úrkomuna stjórna lit. Setjið inn merkingar eins og í lið 1.
4. Nú er sambandið milli árs og hita ekki lengur línulegt, því það kólnaði upp úr 1960 og hlýnaði svo aftur eftir 1990. Reiknið bestu parabólu fyrir gögnin í hægri myndinni (notið `polyfit`) og teiknið hana inn með fallinu `plotpara` sem skilgreint er að ofan.
5. Búið til textareit og setjið inn í hann skilgreiningu á falli P sem þarf að lágmarka til að finna jöfnu bestu parabólu, á svipaðan hátt og fallið S:

   $$\notag S(a,b) = \sum_{i=1}^{n}(ax_i + b - y_i)^2 $$

   er lágmarkað til að finna jöfnu bestu línu.

In [164]:
```
from urllib.request import urlopen
f = urlopen('https://cs.hi.is/strei/hiti-urkoma.txt')
S = np.loadtxt(f)
ar = S[:,0]
h = S[:,1]
u = S[:,-1]

plt.scatter(u, h, c=ar, s=100, edgecolor='darkblue')
plt.colorbar(label='Ár'); plt.grid()
plt.title('Úrkoma miðað við meðalhita')
plt.xlabel('Úrkoma')
plt.ylabel('Hitastig')

n,m = np.polyfit(u, h, 1)
plotline(n,m)

## 3 - scatter
fig, axes = plt.subplots(nrows=1, ncols=1)
plt.subplot(1,2,1); plt.scatter(h, ar, c=u, s=50)
plt.colorbar(label='Úrkoma'); plt.grid()
plt.title('Hiti miðað við ár')
plt.xlabel('Hitastig')
plt.ylabel('Ár')

## 4 - besta parabóla
(x,y,z) = np.polyfit(h, ar, 2)
plotpara(x,y,z)
```





In [95]:
```
## 2 - Fylgni
(m,p) = st.pearsonr(h, u)
print(p)
## p = 0.05, líklegt að það sé fylgni

## 5 - Ekki viss
```

```
0.05098281317918451
```

In [ ]:

S4.3 Hvernig gekk?

Skrifið örfá orð aftast í þennan reit um hvernig ykkur gekk að leysa verkefnið. Var það tímafrekt? Of þungt eða of létt? Lærdómsríkt? Með hverjum var unnið? Setjið nafnið ykkar undir.

In [ ]:
```
Var í miklum vandræðum að finna aðferð úr námskeiðsefninu sem sýnir hvernig á að se
tja f(x0, x1) dæmi í python og komst því ekki áfram í fyrsta dæminu því ég fékk þan
n hluta ekki til að virka.
En reyndi að gera eins mikið og hægt er án útkomunar.

Sigríður Ösp - sos42
```