Starship robotite lokaliseerimine

Lokaliseerimise ja kaardistamise algoritmid

Natuke rakendusmatemaatikat

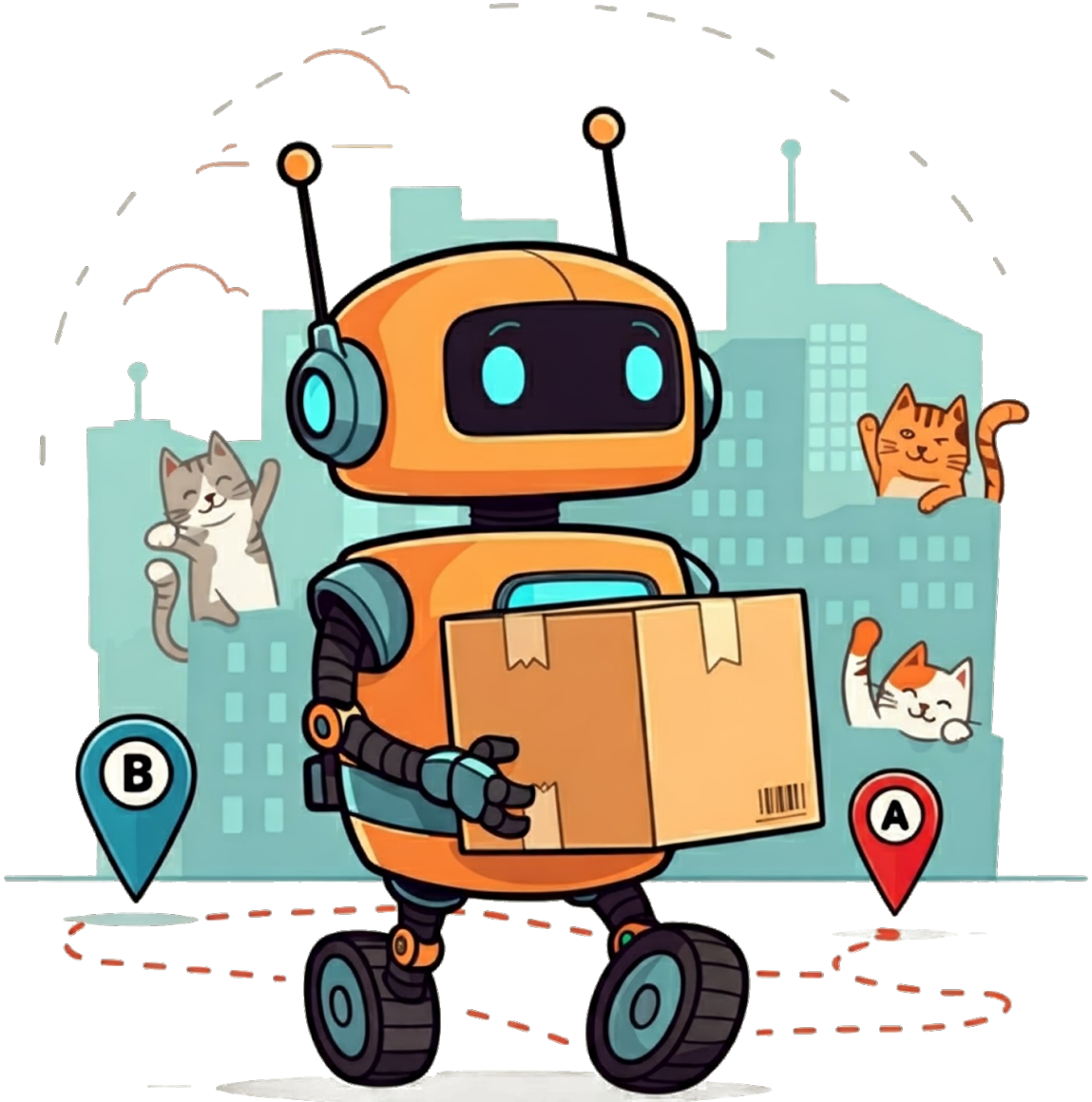
Indrek Mandre

November 25 2024

Pakirobot

Kaardistamise protsess

Kaardistamise protsess



Asukoha täpsus: 50%

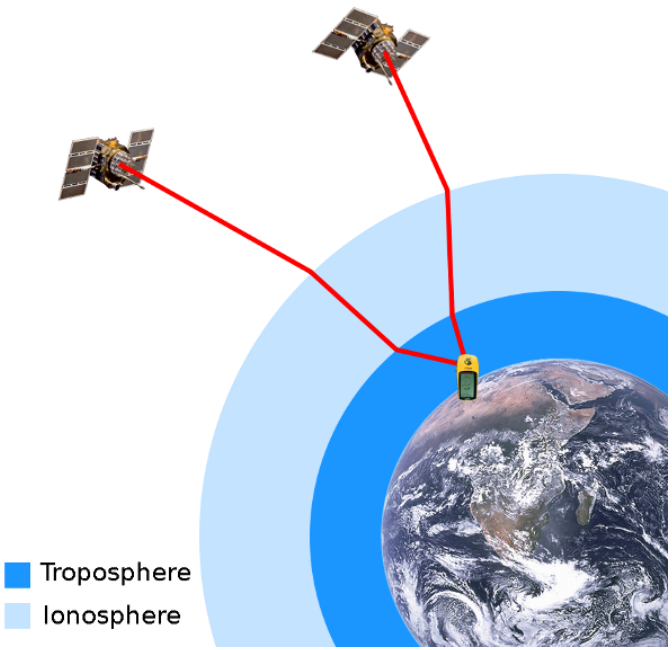
Asukoha täpsus: 100%

Asukoha täpsus: 100%

Lahendus? “The missile knows” meem

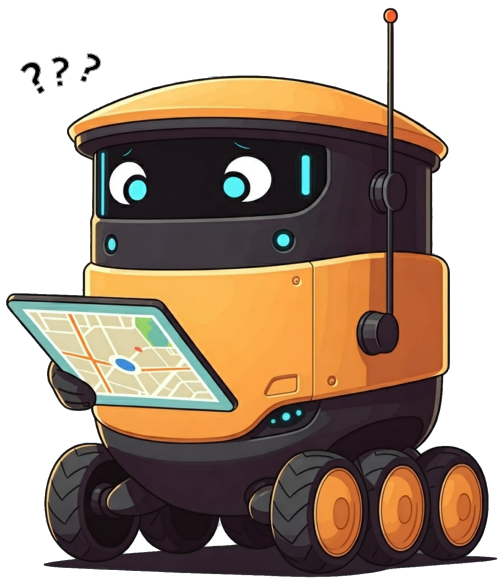
“The missile knows where it is at all times. It knows this because it knows where it isn’t. By subtracting where it is from where it isn’t, or where it isn’t from where it is (whichever is greater), it obtains a difference, or deviation. The guidance subsystem uses deviations to generate corrective commands to drive the missile from a position where it is to a position where it isn’t, and arriving at a position where it wasn’t, it now is. Consequently, the position where it is, is now the position that it wasn’t, and it follows that the position that it was, is now the position that it isn’t ...” – US Air Force

Miks mitte GPS?



Kaardistusalgoritmid: SLAM algoritmid

- Et teada kus me asume, on meil vaja kaarti
- Et märkida kaardile maamärke, peame teadma kus me asume
- Muna ja kana probleem
- Lahendus: Simultaneous Localisation and Mapping (SLAM)
- Kaardistava roboti teekond on osa kaardist

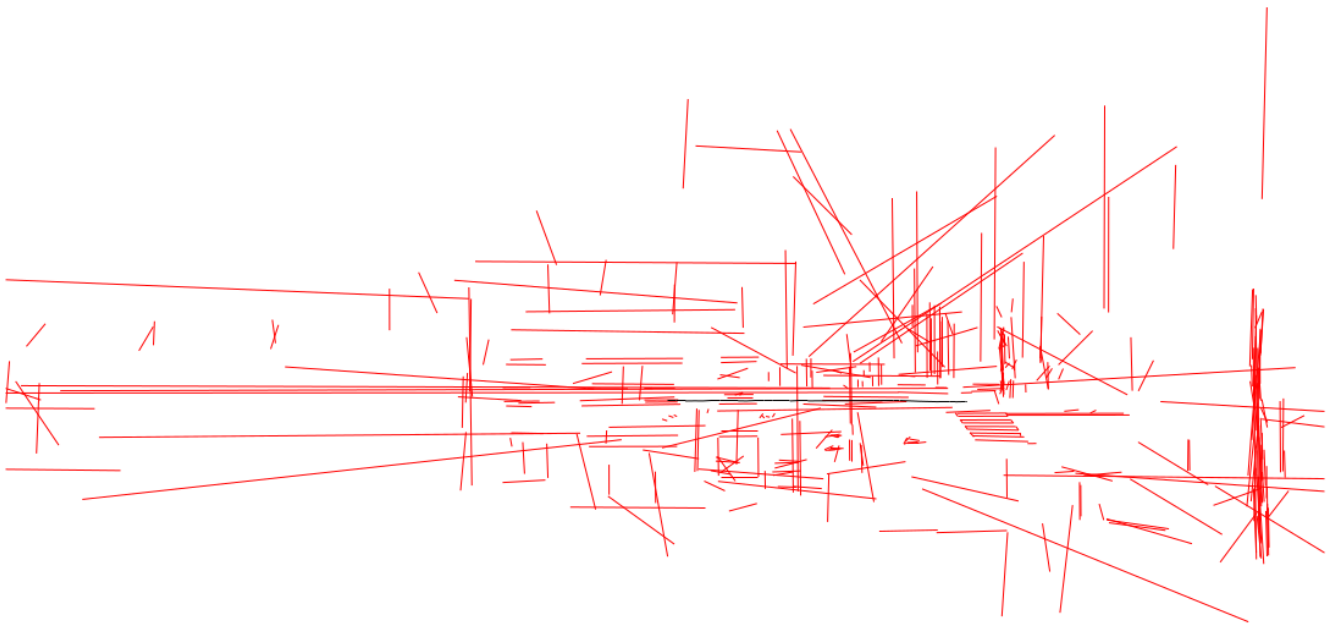


Põhilised sensorid

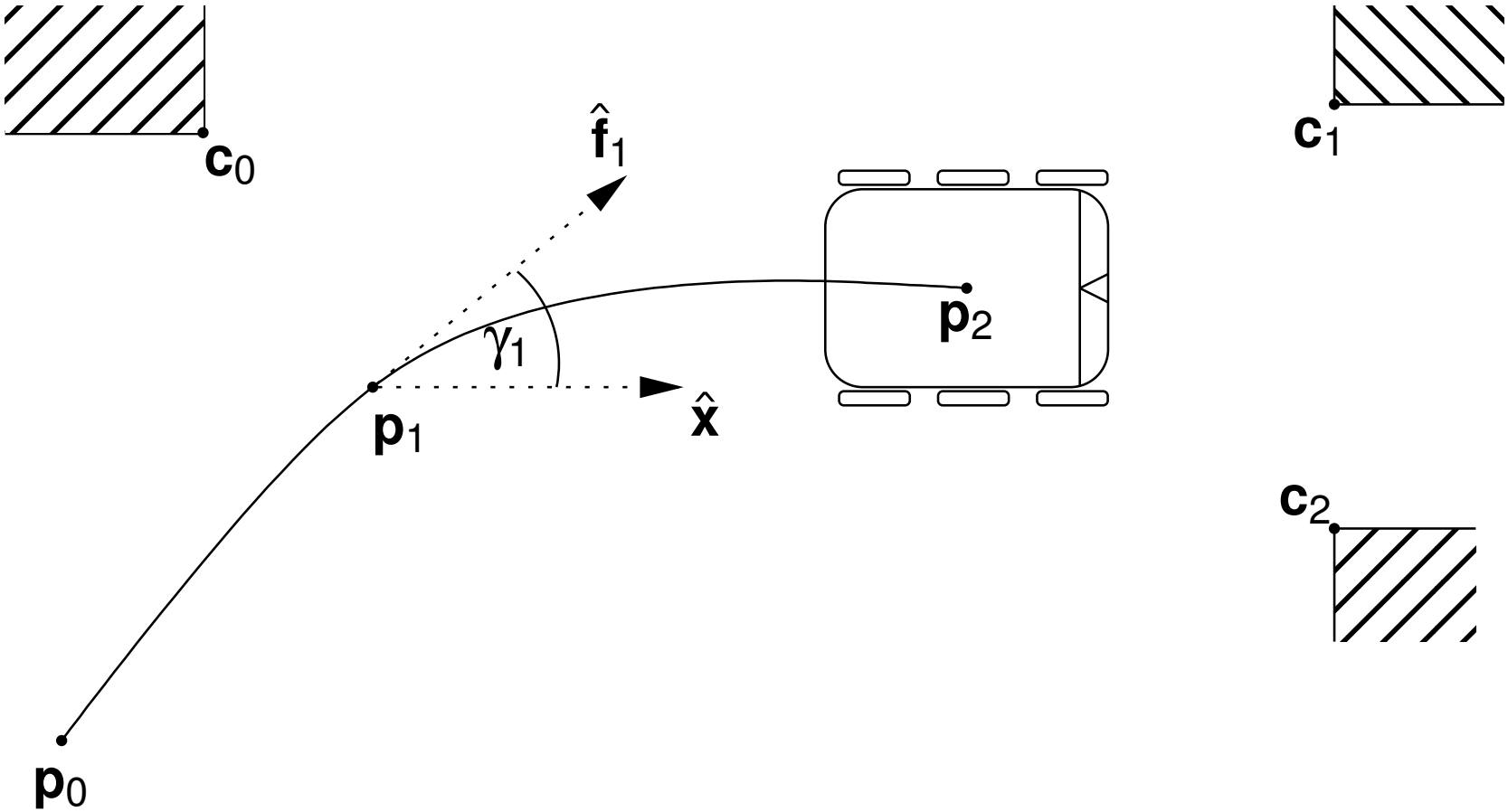
- rataste pöörlemine (odomeetria lugem mootoritest)
- roboti orientatsiooni muutus güroskoobist
- 5 kaamerat

Servade tuvastus

3D kaart



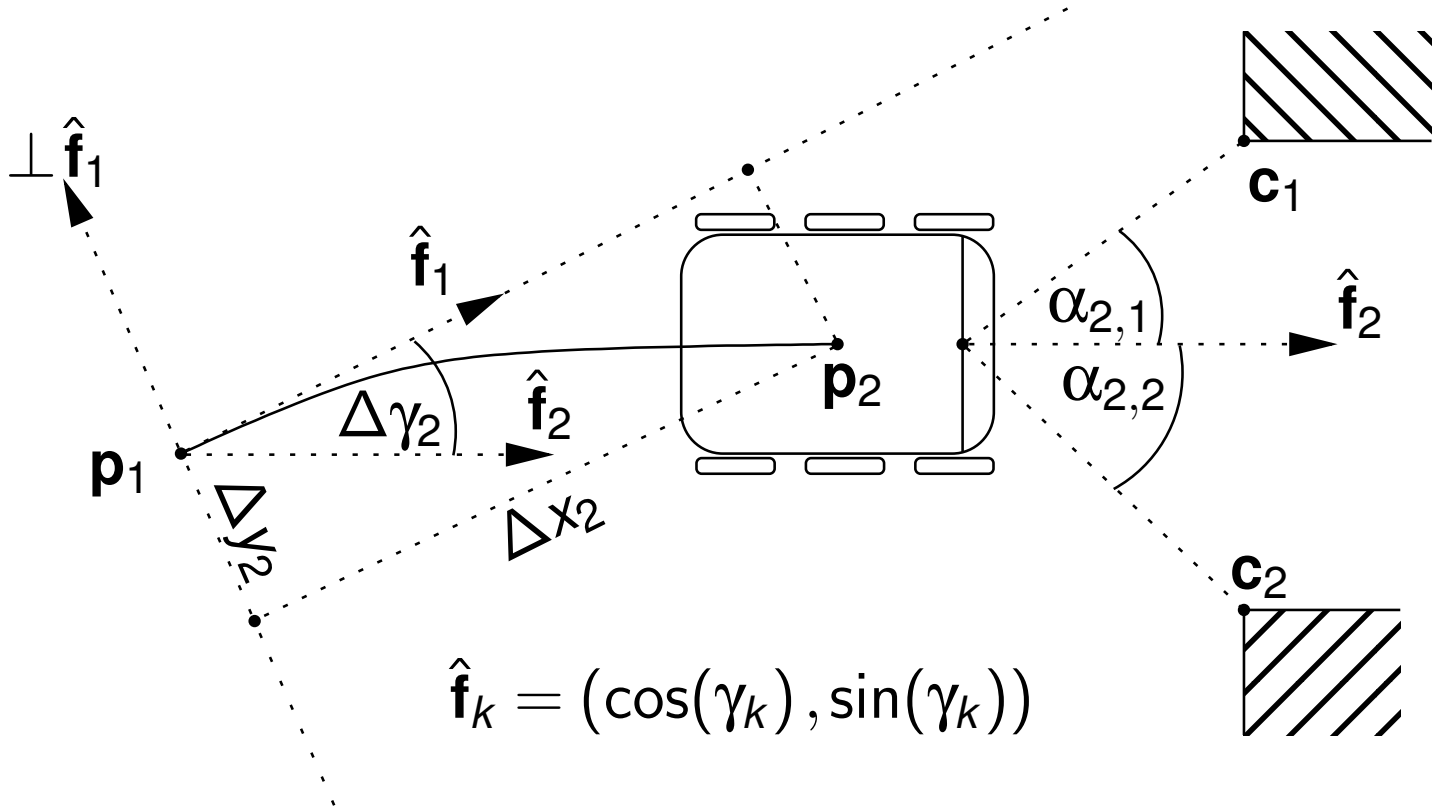
Kaardi mudel



$$\mathbf{p}_k = (x_k, y_k, \gamma_k), \quad \mathbf{c}_k = (c_{k,x}, c_{k,y}),$$

$$M = \left[x_0, y_0, \gamma_0, \ldots, x_N, y_N, \gamma_N, c_{0,x}, c_{0,y}, \ldots, c_{|C|,x}, c_{|C|,y}\right]$$

Sensorite lugemid

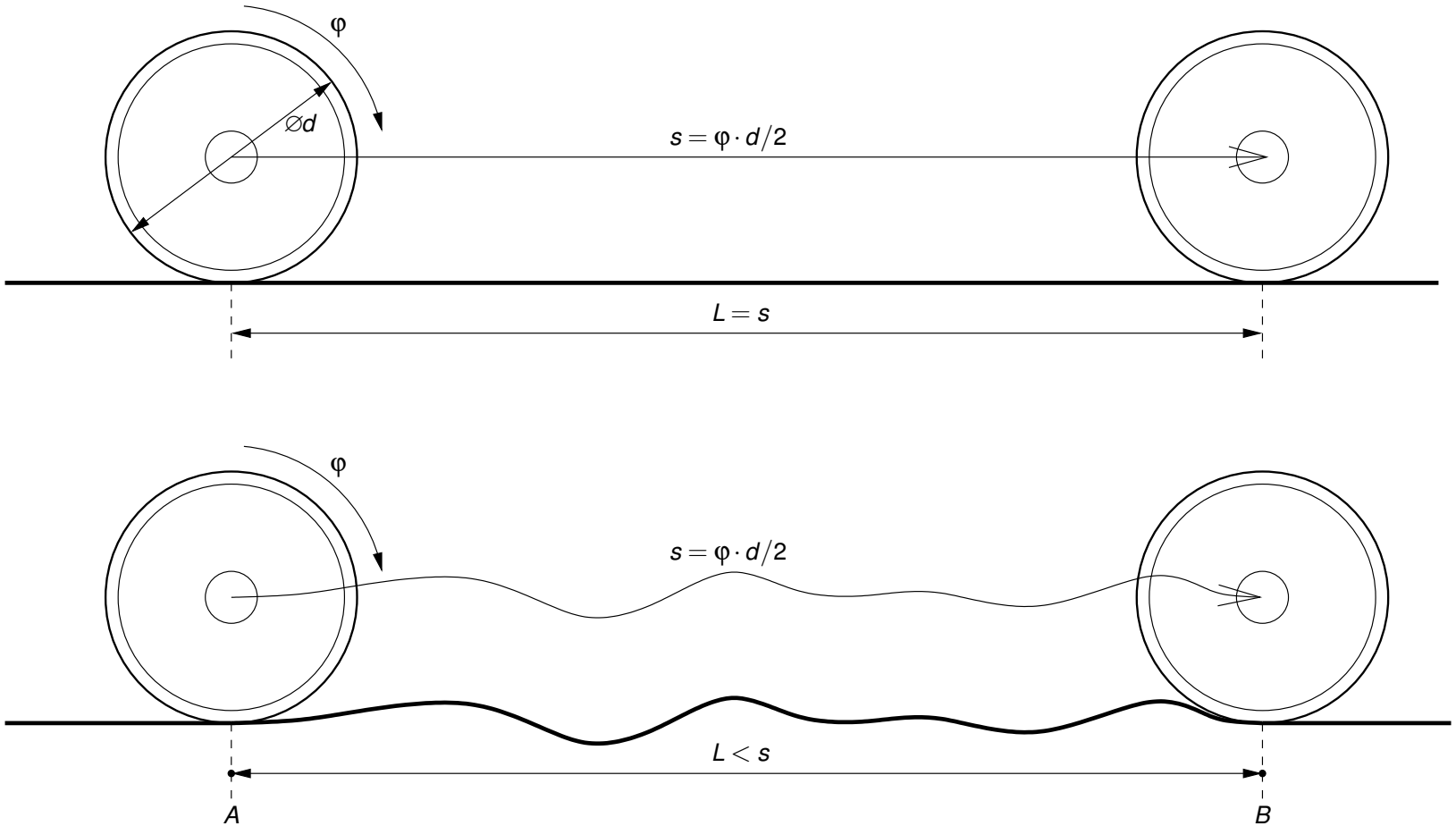


$$\hat{\mathbf{f}}_k = (\cos(\gamma_k), \sin(\gamma_k))$$

$$\Delta\mathbf{p}_2 = (\Delta x_2, \Delta y_2, \Delta\gamma_2), \quad s_{2,1} = (\alpha_{2,1}), \quad s_{2,2} = (\alpha_{2,2}),$$

$$L = [\Delta x_1, \Delta y_1, \Delta\gamma_1, \ldots, \Delta x_N, \Delta y_N, \Delta\gamma_N, \alpha_{i,j}, \ldots]$$

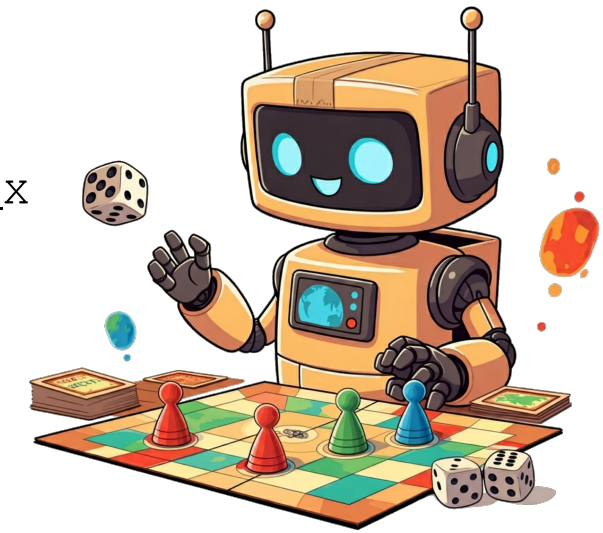
Määramatus I: odomeetria



Määramatus II: liikumise Monte-Carlo mudel

Robot liigub N sammu.

```
1  def generate_path(N, stdev_x, stdev_y):
2      x, y = 0, 0
3      for i in range(0, N):
4          x += 1 + random.uniform(-1, 1)*stdev_x
5          y += random.uniform(-1, 1)*stdev_y
6          yield x, y
7
8  path = generate_path(1000, 0.5, 0.25)
```



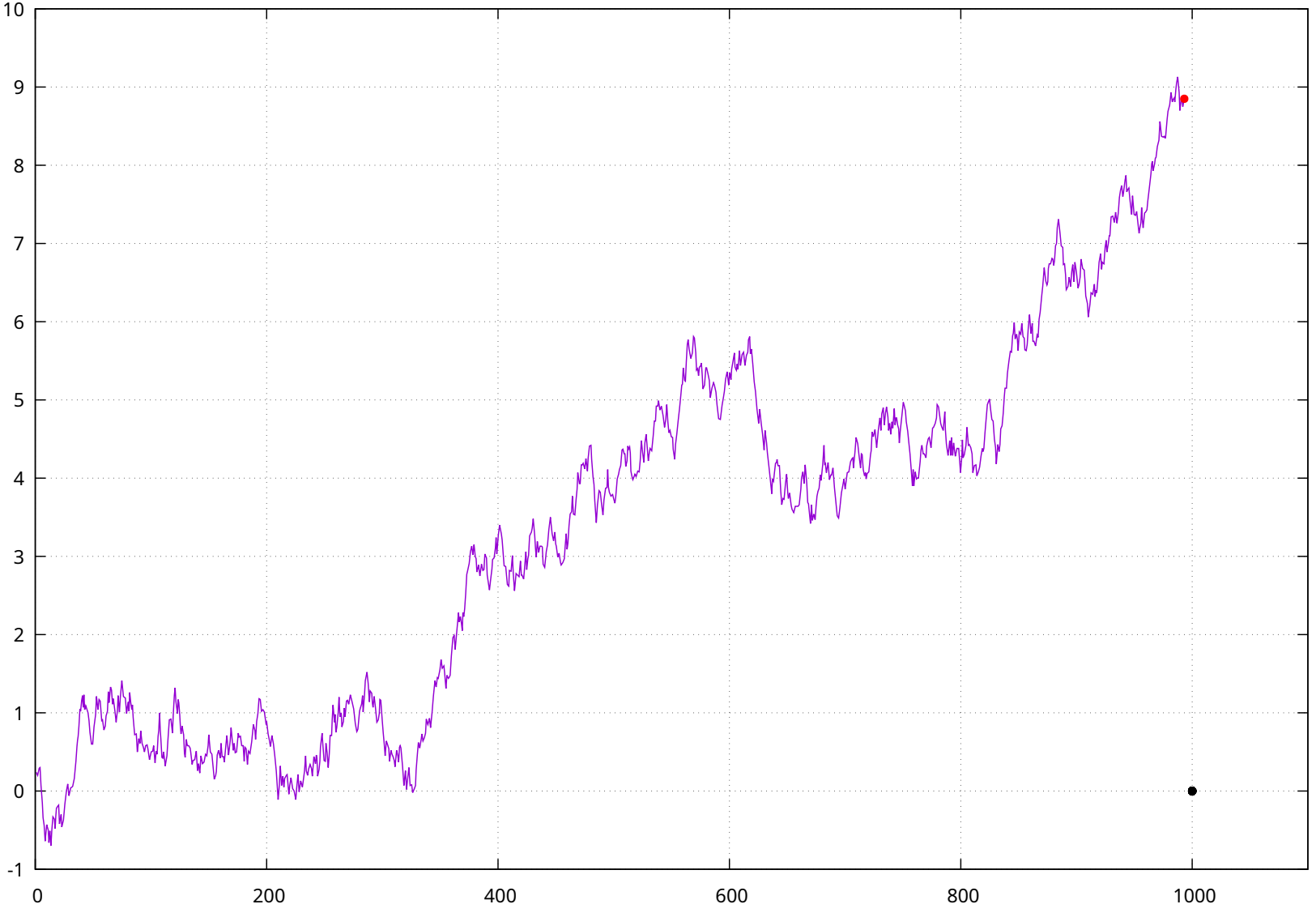
Määramatus III: Monte-Carlo meetodid

- Monte-Carlo: linn kus on palju kasiinosid
- Kasutatakse tihti tänapäeva teaduses kompleksete süsteemide analüüsiks
- Ka arvutustes mis klassikaliselt ei ole praktilised

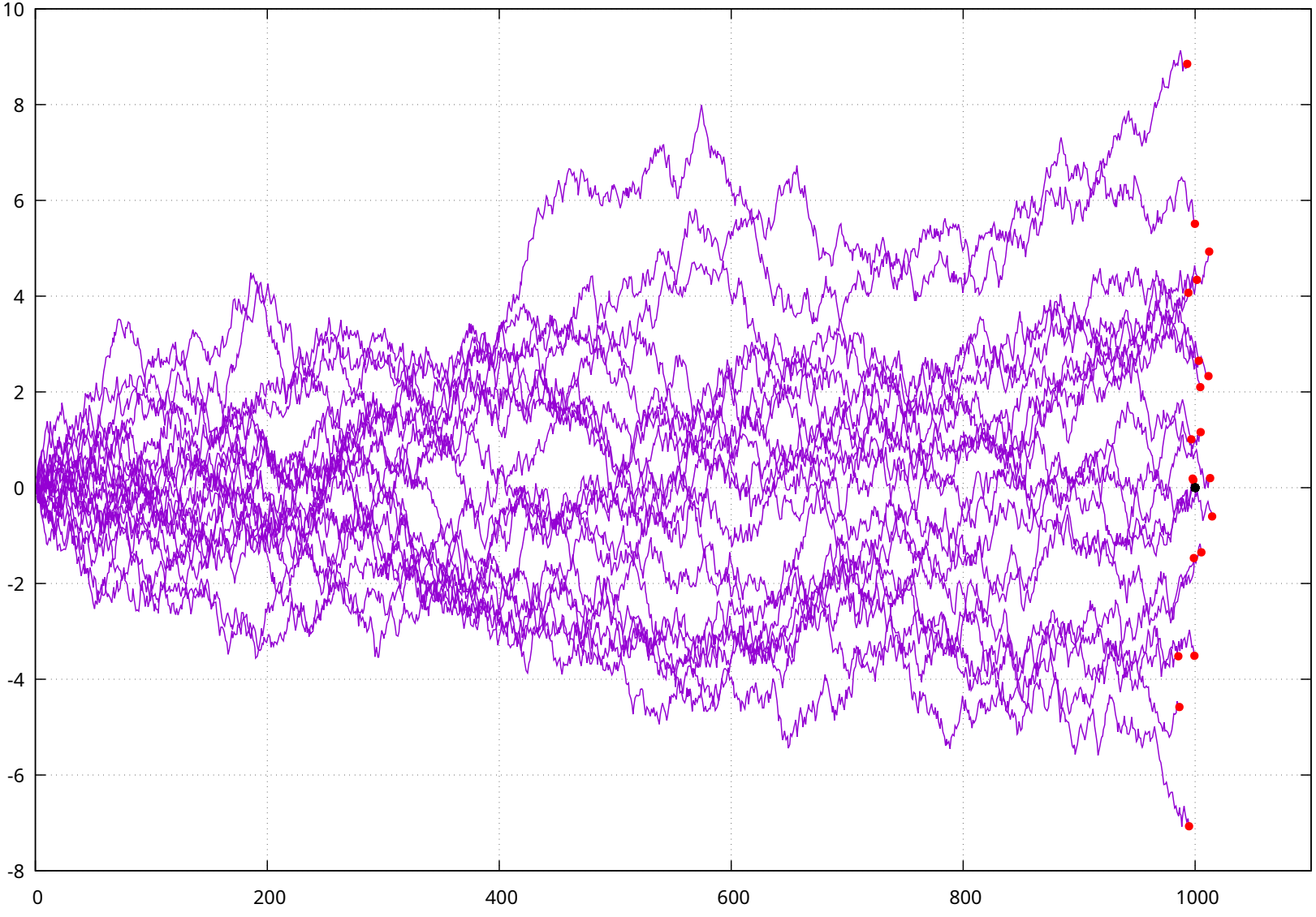
$$\int_V f(x_1, \ldots, x_{1000000})\, dV$$

- Esimesed mudelid 1940-1960 – tuuma- ja vesinikpommi arvutused
  - analüütilise valemi asemel palju “simulatsioone”, mille tulemuste kogumist võetakse üldistused
  - 1953, 1970: Metropolis–Hastings algoritm – kogu süsteemi juhuslik evolutsioon

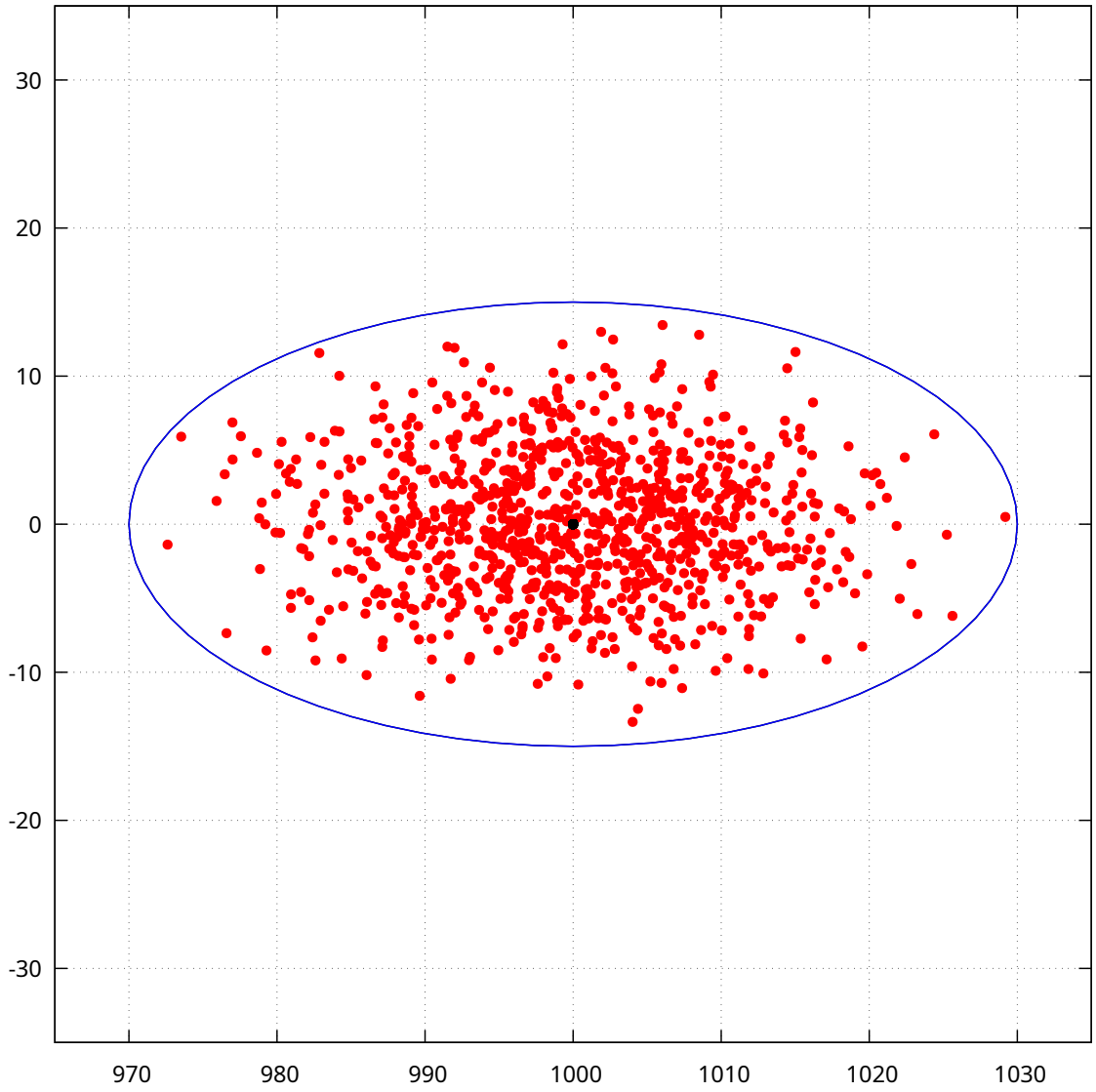
Määramatus IV



Määramatus V



Määramatus VI

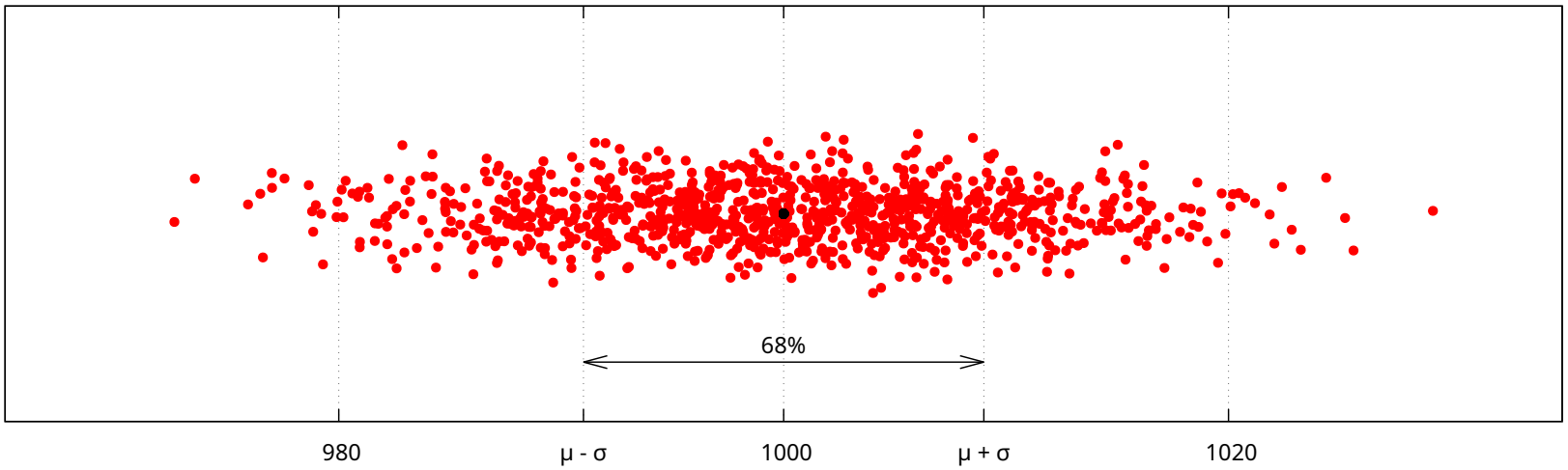


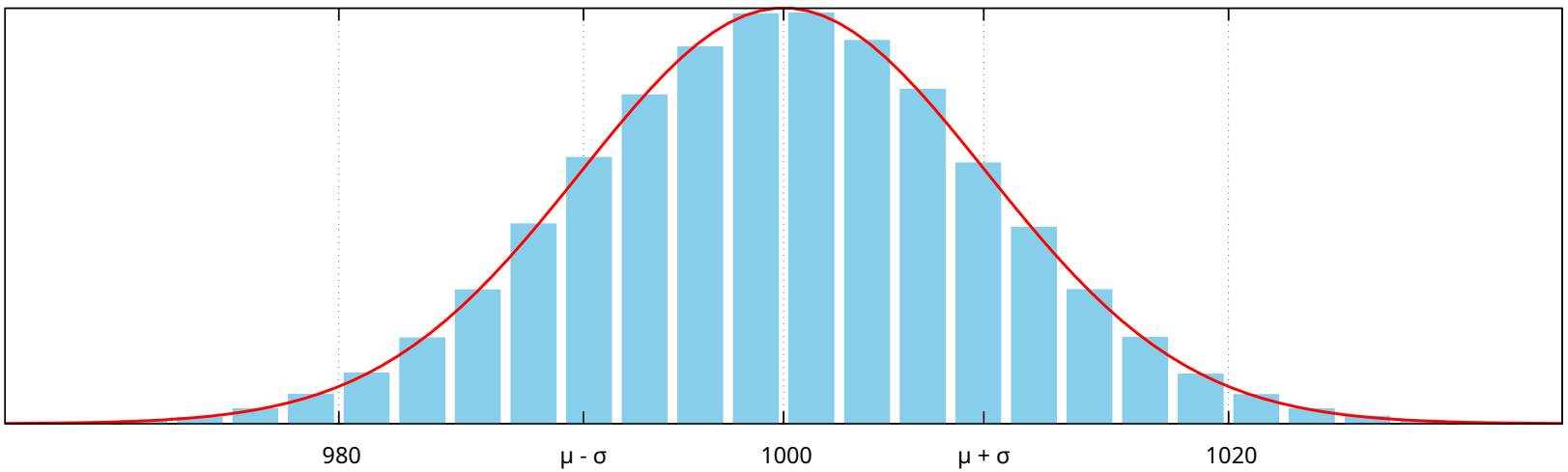
$x = 1000 \pm 30$

$y = 0 \pm 15$

$(p = 99.9\%)$

Määramatus VII: normaaljaotus



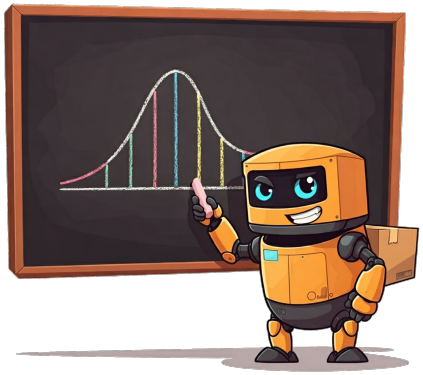


$$x \sim \frac{1}{\sqrt{2\pi\sigma^2}} e^{-\frac{(x-\mu)^2}{2\sigma^2}} = \mathcal{N}(\mu, \sigma^2) \quad [\mu = 1000,\ \sigma \approx 9]$$

Määramatus VIII: tsentraalne piirteoreem

Tsentraalne piirteoreem: olgu

$$Y_N = \sum_{i=1}^{N} X_i,$$



kus X_i on sõltumatud juhuslikud väärtused mille hajuvus on “piiratud”. Siis piiril $n \to \infty$ on Y normaaljaotusega

$$\lim_{N\to\infty} Y_N \sim \mathcal{N}\left(\mu, \sigma^2\right).$$

Teisel kujul:

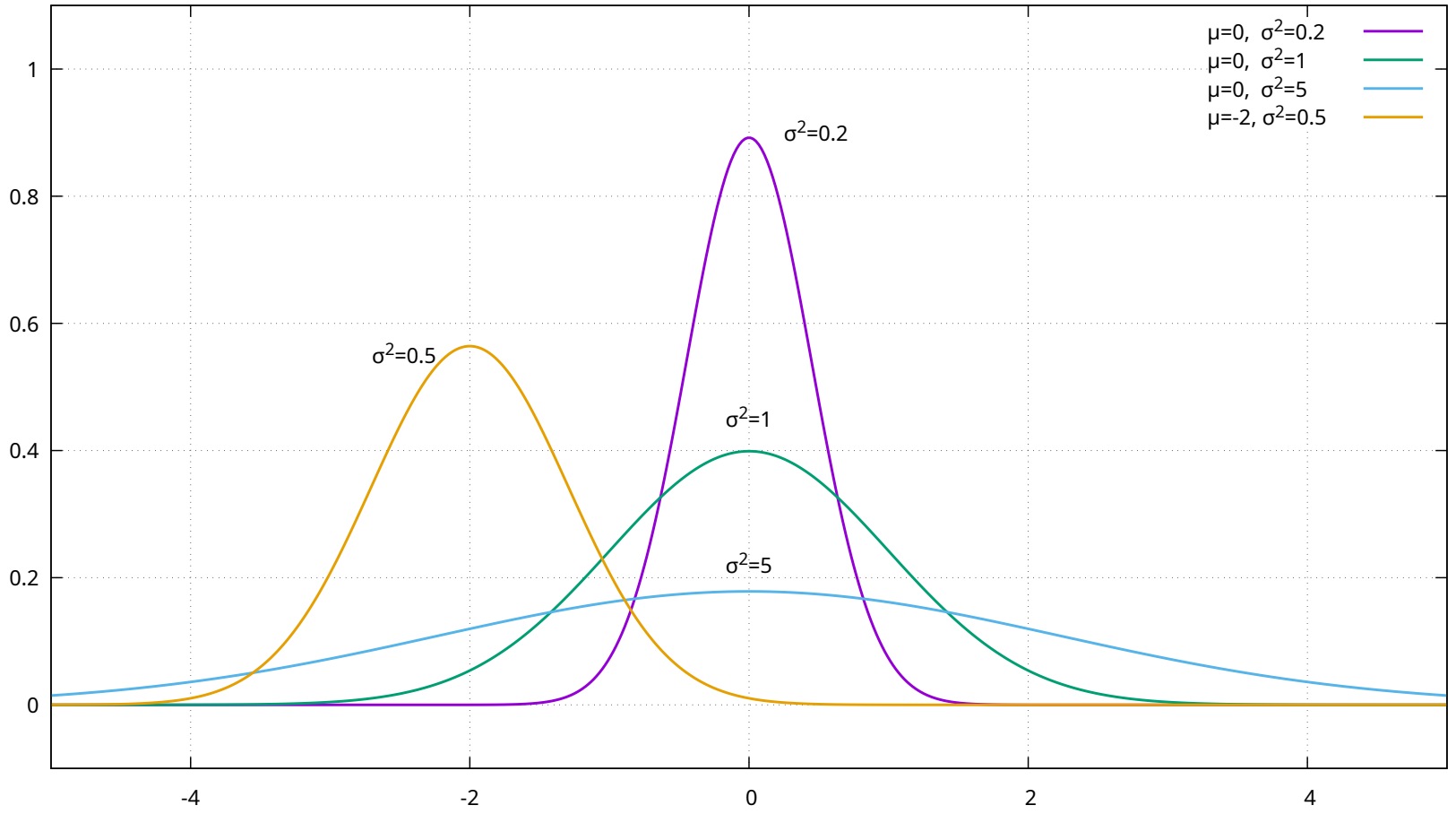
$$\lim_{N\to\infty} \frac{1}{N} \sum_{i=1}^{N} \left(\frac{X_i - \mu_i}{\sigma_i} \right) \sim \mathcal{N}(0,1).$$

Teoreem toimib ka kui X_i on “nõrgalt” seotud. Iga X_i võib olla erineva jaotusega, samas rakendustes tavaliselt $X_i \sim X_j$.

Määramatus IX: näiteid normaaljaotusest

- Roboti odomeetria
- Roboti güroskoop
- Akumuleeriv fotosensor (kaamera piksel)
- Väga paljud sensorid ja mõõteriistad
  - akumuleerivad või korduvalt mõõtvad ja keskmistavad
- Inimeste pikkus, kaal, eluiga, IQ
- Sademete hulk
- Maapinna kõrgus
- Liivatera suurus
- Puulehe suurus

Määramatus X: Dispersioon ja standardhälve



$$\mu \equiv E[X] \approx \frac{1}{N}\sum_{i=1}^{N} X_i = \tilde{\mu},$$

$$\sigma^2 \equiv E\left[(X-\mu)^2\right] = E\left[X^2\right] - E[X]^2 = \mathrm{Var}(X) \approx \frac{1}{N}\sum_{i=1}^{N}(X_i-\tilde{\mu})^2 = \tilde{\sigma}^2.$$

Määramatus XI: kaks ja rohkem dimensiooni

Olgu x normaaljaotusega juhuslik vektor

$$\mathbf{x} = (x_1, \ldots, x_k).$$

Siis tema jaotusfunktsioon on

$$\mathcal{N}(\mathbf{x}, \mathbf{K}) = \frac{1}{(2\pi)^{k/2}\sqrt{|\mathbf{K}|}} e^{-\frac{1}{2}(\mathbf{x}-\boldsymbol{\mu})^T \mathbf{K}^{-1}(\mathbf{x}-\boldsymbol{\mu})}$$

Kovariatsioonimaatriks $\mathbf{K}$ on

$$\mathrm{K} = \begin{bmatrix} \mathrm{Var}(x_1) & \cdots & \mathrm{Cov}(x_1, x_k) \\ \vdots & \ddots & \vdots \\ \mathrm{Cov}(x_k, x_1) & \cdots & \mathrm{Var}(x_k) \end{bmatrix},$$

kus

$$\mathrm{Cov}(X, Y) = E[(X - E(X))(Y - E(Y))].$$

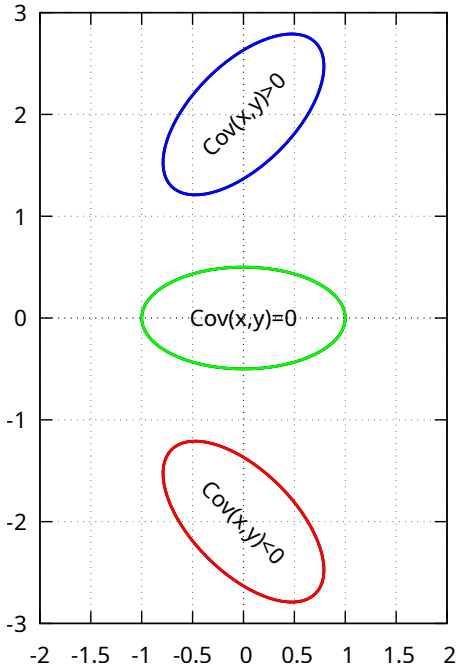
Määramatus XII: kaks ja rohkem dimensiooni

Kui komponentidel pole korrelatsioone,
siis on kovariatsioonimaatriks diagonaalmaatriks:

$$\mathbf{K} = D = \begin{bmatrix} \sigma_1^2 & \cdots & 0 \\ \vdots & \ddots & \vdots \\ 0 & \cdots & \sigma_k^2 \end{bmatrix}$$

ja

$$\mathcal{N}(\mathbf{x}, \mathbf{K}) = \frac{1}{(2\pi)^{k/2}\sqrt{\sum_i \sigma_i^2}} e^{-\frac{1}{2}\sum_i \frac{(x_i-\mu_i)^2}{\sigma_i^2}}.$$

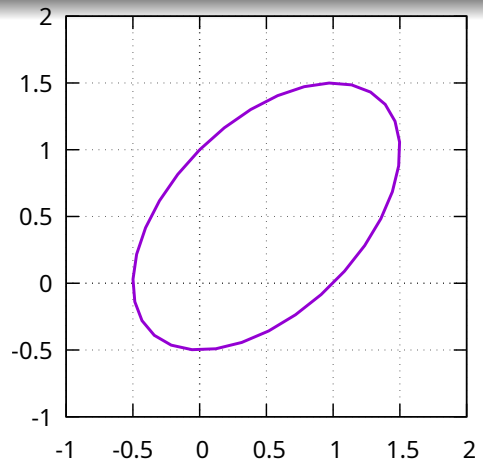


Teine perspektiiv: võib leida koordinaatsüsteemi,
kus muutujatel pole korrelatsioone:

$$\mathbf{K} = RDR^{-1} = RDR^T.$$

Määramatus XIII: ellipsi joonistamine

```
1  from sympy import Matrix, cos, sin, \
2    sqrt, pi
3  import numpy as np
4  def cov_ellipse(mu, K):
5    # R koosneb omavektoritest
6    # D omaväärtustest
7    R, D = K.diagonalize()
8    assert (R*D*R.transpose() - K).norm() < 1e-10
9    stdev_x, stdev_y = sqrt(D[0, 0]), sqrt(D[1, 1])
10   for phi in np.linspace(0, float(2*pi), 32):
11     pos_ellipse = Matrix([[stdev_x * cos(phi)],
12                           [stdev_y * sin(phi)]]).evalf()
13     pos_world = mu + R * pos_ellipse
14     yield pos_world[0], pos_world[1]
15 mu = Matrix([0.5, 0.5])
16 K = Matrix([
17   [1.0, 0.5],
18   [0.5, 1.0]]
19 )
20 plot(cov_ellipse(mu, K))
```



Määramatus XIV: inseneri vaade

Rakendustes käsitletakse tihti kõiki juhuslikke väärtusi nagu nad oleksid normaaljaotusega

- standardhälve kui hinnang vea ülempiirile
- kompaktne esitlus: ainult kaks parameetrit μ ja σ
- lihtne analüütiliselt töödelda

Üldistus mis võib viia ka halbade tagajärgedeni:

- multimodaalsed jaotused
- jaotusel on asümmeetriline “pikk saba”

Sensorite lugem: kasutame haamrit

Meil oli sensorite lugem

$$L = [\Delta x_1, \Delta y_1, \Delta \gamma_1, \ldots, \Delta x_P, \Delta y_P, \Delta \gamma_P, \alpha_{0,0}, \ldots] .$$

L on normaaljaotusega:

$$L \sim \mathcal{N}(\boldsymbol{\mu}_L, \mathbf{K}_L) .$$

$\mathbf{K}_L$ on plokk-diagonaalne maatriks.

$$\mathbf{K}_L = \begin{bmatrix} K_1 & \cdots & \mathbf{0} \\ \vdots & \ddots & \vdots \\ \mathbf{0} & \cdots & K_m \end{bmatrix} .$$

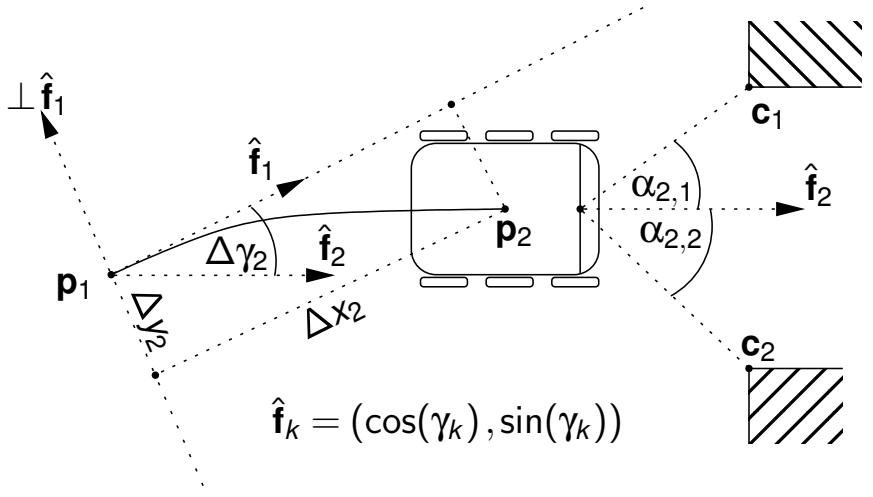
Kui meil oleks tõene kaart M_{GT}, siis

$$\boldsymbol{\mu}_L = \left(F_0(M_{\mathrm{GT}}), \ldots, F_{|L|}(M_{GT})\right) .$$

Funktsioonide $F_j(M)$ koostamine: näide

```
1   from sympy import symbols, Matrix, sin, cos
2
3   def rotation_matrix(alpha):
4     return Matrix([[cos(alpha), -sin(alpha)],
5                    [sin(alpha),  cos(alpha)]])
6
7   def pose_delta(pos0, gamma0, pos1, gamma1):
8     mr2w = rotation_matrix(gamma0)
9     r = mr2w.transpose() * (pos1 - pos0)
10    return r[0], r[1], gamma1 - gamma0
11
12  x1, y1, g1, x2, y2, g2 = \
13    symbols('x1 y1 g1 x2 y2 g2')
14
15  delta_x2, delta_y2, delta_g2 = pose_delta(
16      Matrix([[x1], [y1]]), g1,
17      Matrix([[x2], [y2]]), g2
18  )
19  print(f'delta_x2 = {delta_x2}')
20  print(f'delta_y2 = {delta_y2}')
21  print(f'delta_g2 = {delta_g2}')
22
23  value_map = {x1: 0, y1: 0, g1: 0,
24    x2: 10, y2: 0, g2: 0.1}
25
26  print(f'delta_x2 = {delta_x2.subs(value_map).evalf():.2f}')
27  print(f'delta_y2 = {delta_y2.subs(value_map).evalf():.2f}')
28  print(f'delta_g2 = {delta_g2.subs(value_map).evalf():.2f}')
```

```
delta_x2 = (-x1 + x2)*cos(g1) + (-y1 + y2)*sin(g1)
delta_y2 = -(-x1 + x2)*sin(g1) + (-y1 + y2)*cos(g1)
delta_g2 = -g1 + g2
delta_x2 = 10.00
delta_y2 = 0.00
delta_g2 = 0.10
```



$$M = \left[x_0, y_0, \gamma_0, \ldots, x_N, y_N, \gamma_N, c_{0,x}, c_{0,y}, \ldots, c_{|C|,x}, c_{|C|,y}\right]$$

$$L = \left[\Delta x_1, \Delta y_1, \Delta\gamma_1, \ldots, \Delta x_P, \Delta y_P, \Delta\gamma_P, \alpha_{0,0}, \ldots\right]$$

Bayesi teoreem

Bayesi teoreem:

$$P(M|L) = \frac{P(L|M)\,P(M)}{P(L)}.$$

Mudeli parameetrid M on muutujad ja sensorite lugem L on konstant.

Me otsime “parimat” kaarti, mille tõenäosus on suurim:

$$M_{BEST} = \arg\max_{M \in \mathbb{M}} P(M|L)\,.$$

Bayesi teoreem II

Sensorid on konstant selles valemis, ehk

$$P(L) = const.$$

Meil on juba mingi kaardi algväärtus ja me otsime seal ümbruses olevate kaartide hulgast, mis on piiratud meie mudeliga. Me väidame et

$$P(M) \approx const. \qquad M \in \mathbb{M}_{search} \subset \mathbb{M}.$$

Järgi jääb:

$$P(M|L) \simeq C \cdot P(L|M)\,.$$

Aga

$$L|M \sim \mathcal{N}(\boldsymbol{\mu}(M), K_L)\,,$$

siis

$$P(M|L) \simeq Ce^{-\frac{1}{2}(L-\boldsymbol{\mu}(M))^T K_L^{-1}(L-\boldsymbol{\mu}(M))}.$$

Lahendus

Kuna $\ln e^x = x$ ja $\ln a \cdot b = \ln a + \ln b$, siis

$$\ln P(M|L) \simeq lnC - \frac{1}{2}(S - \mu(M))^T K_L^{-1}(L - \mu(M)).$$

Et leida parim kaart, tuleb lahendada minimeerimisülesanne

$$M_{BEST} = \underset{M \in \mathbb{M}_{search}}{\arg\min} (L - \mu(M))^T K_L^{-1}(L - \mu(M)).$$

Kui K_L on diagonaalne, taandub see mittelineaarseks vähimruutude meetodiks:

$$M_{BEST} = \underset{M \in \mathbb{M}_{search}}{\arg\min} \sum_i \frac{(L_i - F_i(M))^2}{\sigma_i^2}.$$

Lokaliseerimine, üldistus

Lokaliseerimisel kaardi maamärgid pole enam osa mudeli parameetritest vaid on fikseeritud:

$$\begin{aligned} M_L &= [x_0, y_0, \gamma_0, \ldots, x_N, y_N, \gamma_N], \\ L_L &= L = [\Delta x_1, \Delta y_1, \Delta \gamma_1, \ldots, \Delta x_P, \Delta y_P, \Delta \gamma_P, \alpha_{0,0}, \ldots]. \end{aligned}$$

Kas meetod/mudel on piiratud robotitega?

- Lihtne lineaarne regressioon: $y = ax + b$, parameetrid $[a, b]$
- füüsikaseaduste parameetrite leidmiseks

Vaatame hiljem ka

- kuidas hinnata mudeli korrektsust (kas mudel üldse vastab reaalsusele)
- leitud parameetrite määramatus

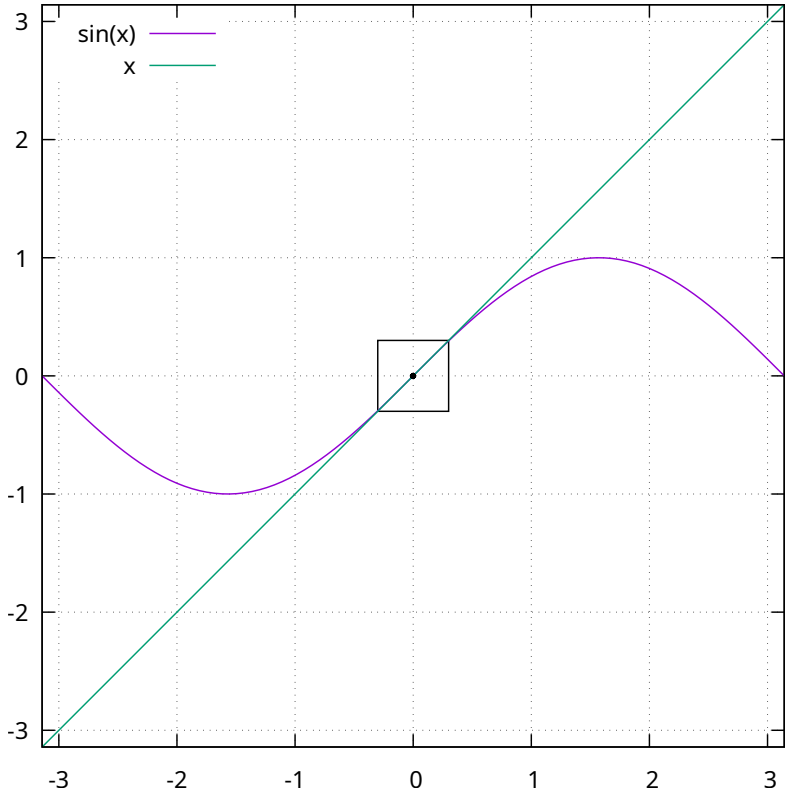
$$M \sim \mathcal{N}(\boldsymbol{\mu}_M, \mathbf{K}_M)$$

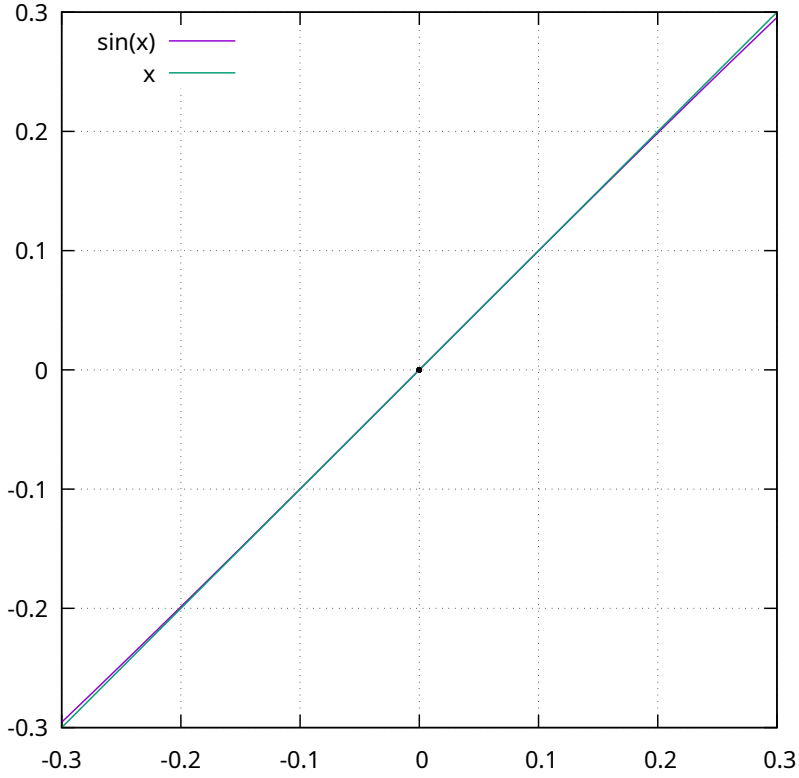
Lineariseerimine I

$$\sin(0.1) \approx ?$$

$$0.1 \cancel{\text{rad}} \cdot \frac{180^\circ}{\pi \cancel{\text{rad}}} = 5.7296^\circ$$

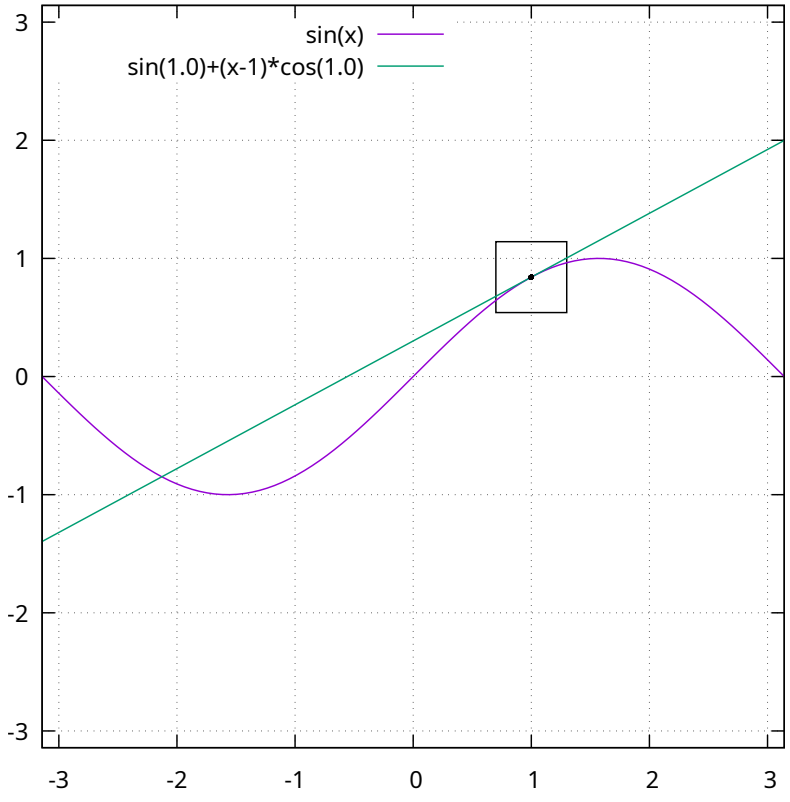
Lineariseerimine II

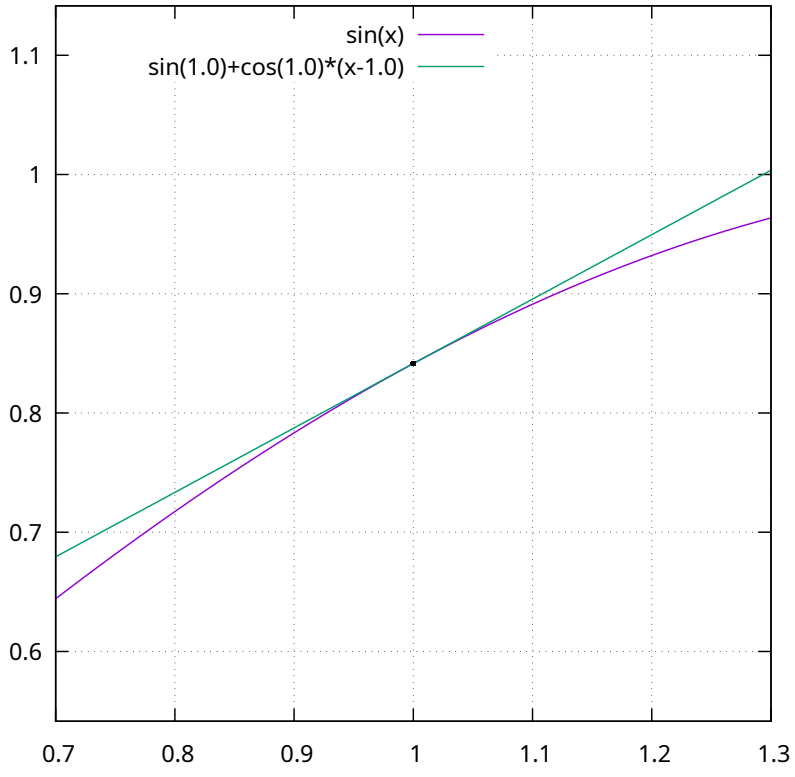




$$\begin{aligned}
\sin(0.1) &\approx 0.1, \\
\sin(x) &\approx x, \qquad |x| \ll 1 \\
0.1 - \sin(0.1) &= 0.000167, \\
\frac{0.1 - \sin(0.1)}{\sin(0.1)} \cdot 100\% &= 0.17\%
\end{aligned}$$

Lineariseerimine III



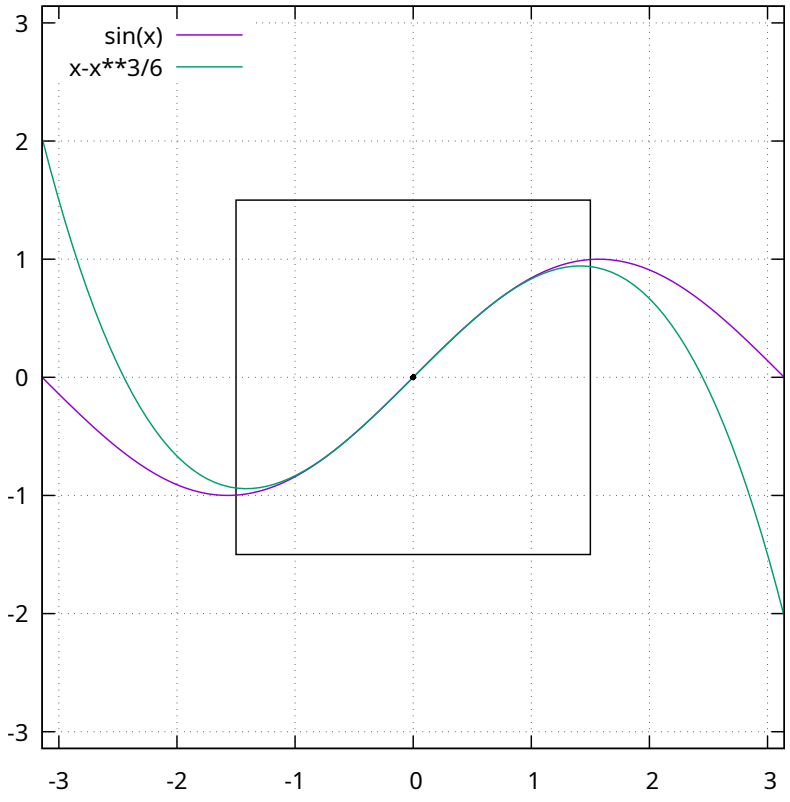


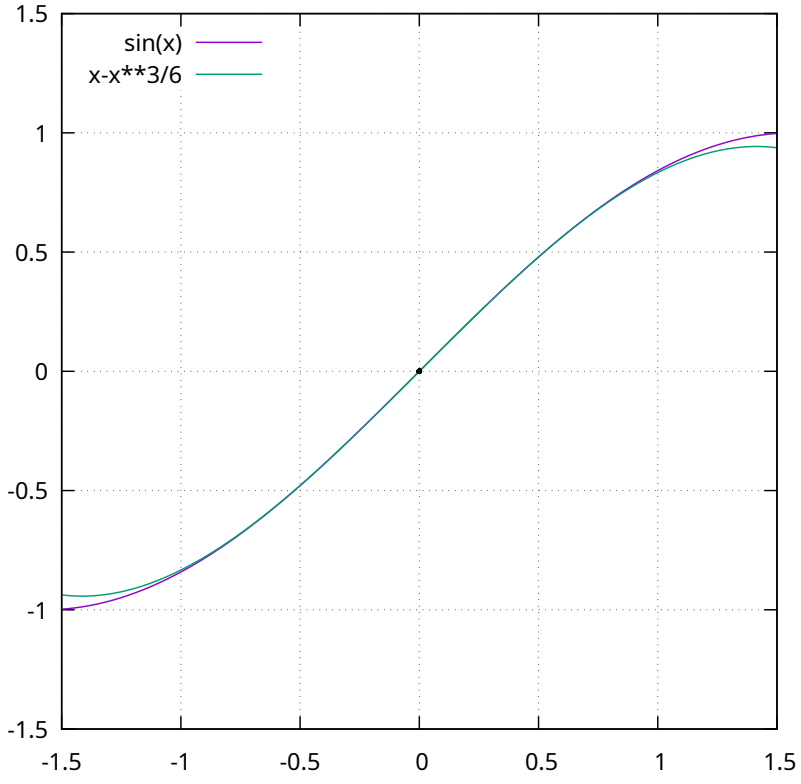
$$\sin(x) \approx \sin(x_0) + \cos(x_0)(x - x_0), \qquad |x - x_0| \ll 1$$

$$\sin(1+\delta) \approx \sin(1) + \cos(1)\cdot\delta = 0.8415 + 0.5403\cdot\delta, \qquad |\delta| \ll 1$$

$$f(x) \approx f(x_0) + f'(x_0)(x - x_0), \qquad |x - x_0| \ll c_f;\ f' \equiv \frac{df}{dx}$$

Taylori rida I



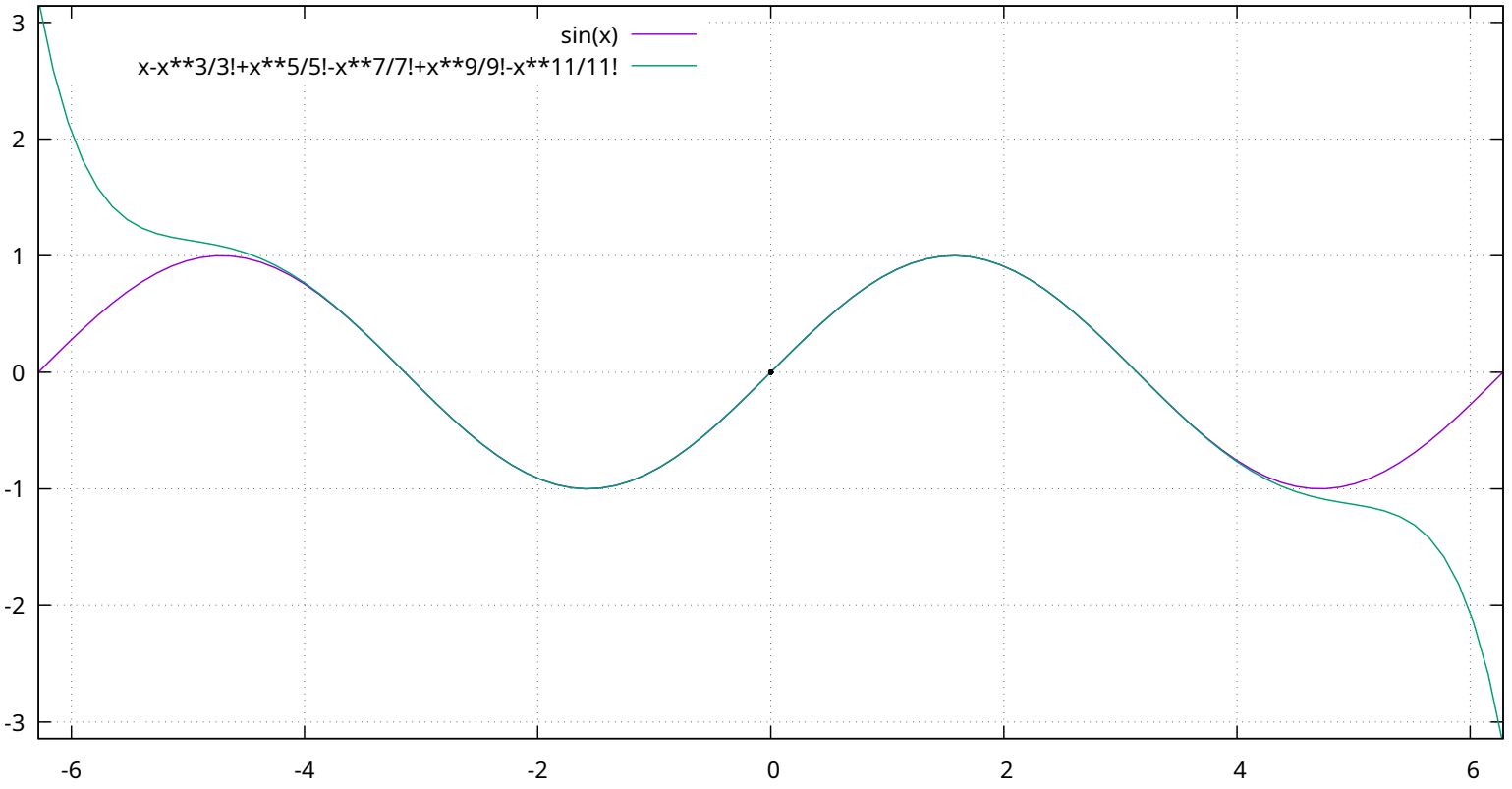


$$\sin(x) \approx x - \frac{1}{6}x^3, \qquad |x| \ll 1$$

$$\sin(0.1) \approx 0.099833,$$

$$\frac{\sin(0.1) - 0.099833}{\sin(0.1)} \cdot 100\% = 0.000083\%$$

Taylori rida II



$$\sin(x) = x - \frac{x^3}{3!} + \frac{x^5}{5!} - \frac{x^7}{7!} + \frac{x^9}{9!} - \ldots = \sum_{n=0}^{\infty} \frac{x^{2n+1}}{(2n+1)!}$$

Otsime

$$f(x) = f(x_0) + a(x - x_0) + b(x - x_0)^2 + c(x - x_0)^3 + \ldots$$

Taylori rida III

$$
\begin{aligned}
f(x) &= f(x_0) + f'(x_0)(x - x_0) + \frac{f''(x_0)}{2!}(x - x_0)^2 + \dots \\
&= \sum_{i=0}^{\infty} \frac{f^{(i)}(x_0)}{i!}(x - x_0)^i, \\
f(x) &= \sum_{i=0}^{n} \frac{f^{(i)}(x_0)}{i!}(x - x_0)^i + R_n(x).
\end{aligned}
$$

Ei pruugi kehtida igas punktis. Esimesed liikmed on olulisemad tänu kordajatele:

$$\lim_{n\to\infty} \frac{1}{n!} = 0.$$

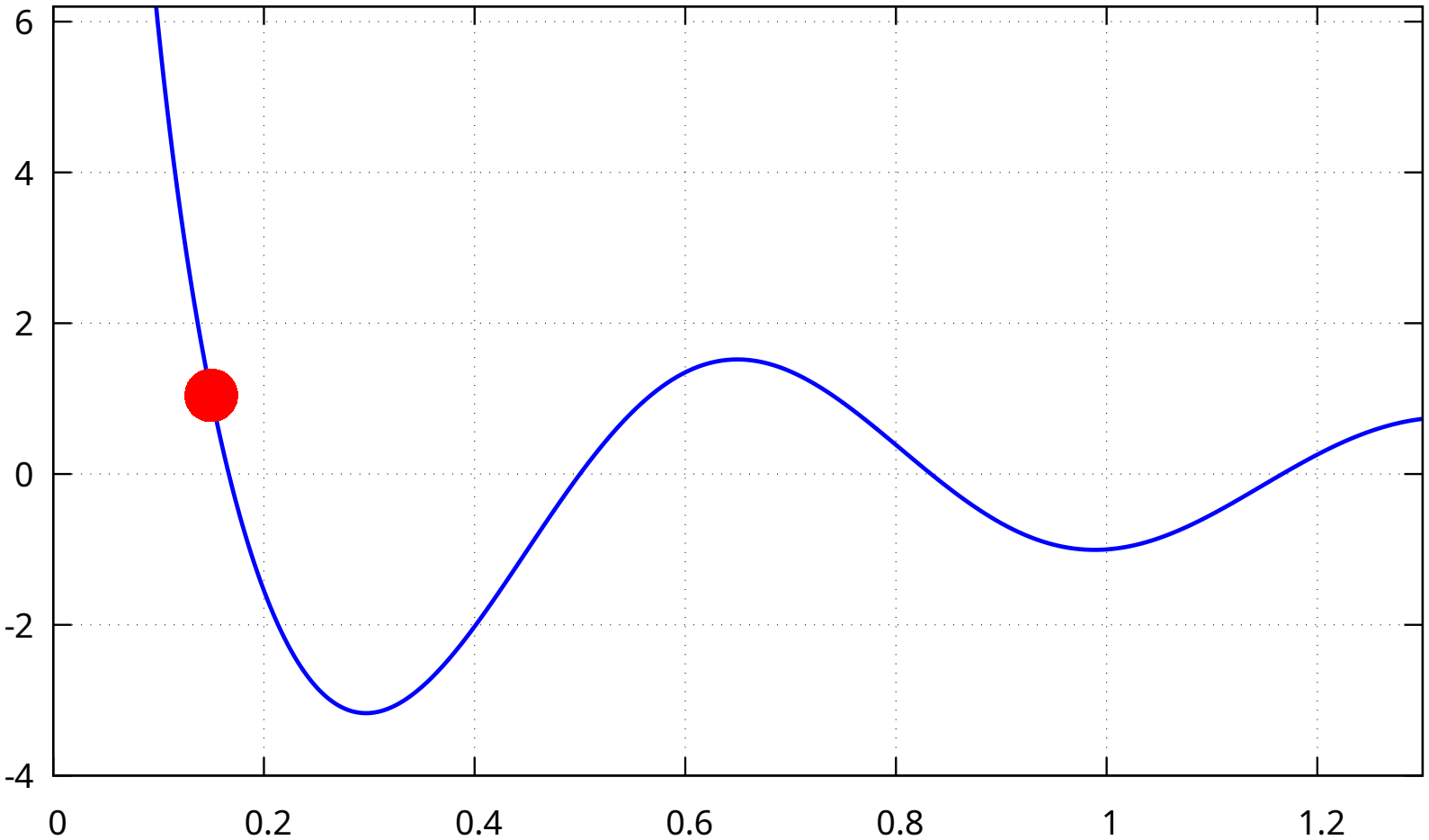
Näiteks $1/6! \approx 0.001$. Miks x_0 ligidal on tulemus täpsem:

$$|x - x_0| \ll 1 \Rightarrow \lim_{n\to\infty} (x - x_0)^n = 0.$$

Näiteks $0.1^3 = 0.001$.

Mittelineaarse optimeerimise meetodid

$$r_i = \frac{L_i - F_i(M)}{\sigma_i}, \quad S(M) = \sum_i r_i^2.$$



Gradiendimeetodid: kiireima laskumise meetod

Funktsiooni ekstreemum on seal kus tõus on null:

$$\frac{df}{dx} = 0.$$

Mitme muutuja funktsiooni korral:

$$\nabla S = \left[\frac{\partial S}{\partial m_1}, \ldots, \frac{\partial S}{\partial m_{|M|}} \right],$$

$$\nabla S = \mathbf{0}.$$

Gradiendi suund on funktsiooni suurima kasvu suunas antud punktis.
Järelikult tuleb liikuda vastupidises suunas:

$$\begin{aligned} M_k &= M_{k-1} - \lambda_k \cdot \nabla S(M_{k-1}), \\ \lambda_k &= \arg\min_{\lambda \in (0,\infty)} S(M_{k-1} - \lambda \cdot \nabla S(M_{k-1})). \end{aligned}$$

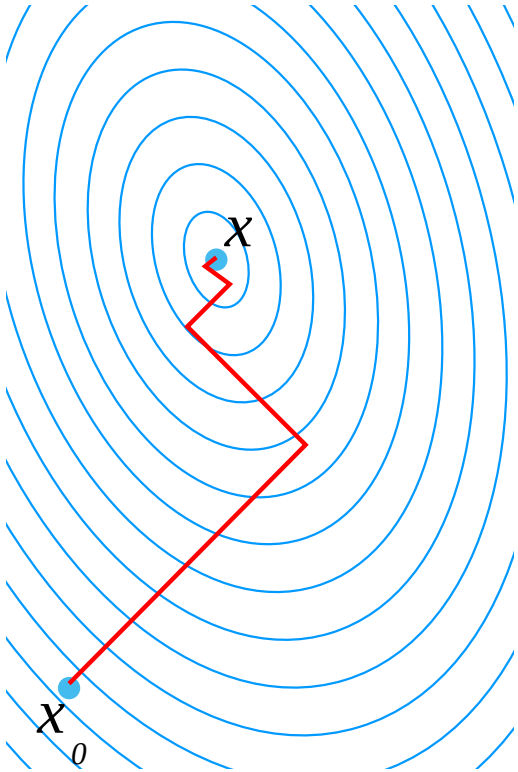
Korrata kuni

$$\|\nabla S(M_k)\| < \varepsilon.$$

Kaasgradientide meetod: siksakitamise vastu

Idee: liikuda risti eelmise liikumise suunaga.

$$\begin{aligned}
\mathbf{x}\cdot\mathbf{y} &= \cos\theta\,\|\mathbf{x}\|\,\|\mathbf{y}\|,\\
\mathbf{x}_{\perp\mathbf{y}} &= \mathbf{x}-\frac{\mathbf{x}\cdot\mathbf{y}}{\|\mathbf{y}\|^2}\mathbf{y},\\
\left(\mathbf{x}-\frac{\mathbf{x}\cdot\mathbf{y}}{\|\mathbf{y}\|^2}\mathbf{y}\right)\cdot\mathbf{y} &= \mathbf{x}\cdot\mathbf{y}-\mathbf{x}\cdot\mathbf{y}=0.
\end{aligned}$$



Algoritm:

$$\begin{aligned}
\mathbf{g}_k &= -\nabla S(M_{k-1})\\
\mathbf{s}_k &= \mathbf{g}_k+\beta(\mathbf{g}_k,\mathbf{g}_{k-1},\mathbf{s}_{k-1})\,\mathbf{s}_{k-1}\\
M_k &= M_{k-1}+\lambda_k\cdot\mathbf{s}_k,\\
\lambda_k &= \underset{\lambda\in(0,\infty)}{\arg\min}\, S(M_{k-1}-\lambda\cdot\mathbf{s}_k).
\end{aligned}$$

Polak-Ribiere:

$$\beta(\mathbf{g}_k,\mathbf{g}_{k-1},\mathbf{s}_{k-1})=\frac{\mathbf{g}_k^T(\mathbf{g}_k-\mathbf{g}_{k-1})}{\mathbf{g}_{k-1}^T\mathbf{g}_{k-1}}.$$

Gauss-Newtoni meetod

Aproksimeerime funktsiooni $S(M)$ kasutades Taylori rea kolme esimest liiget:

$$\widetilde{S}(M_{k-1}+\Delta_k) = S(M_{k-1}) + \Delta_k^T \nabla S(M_{k-1}) + \frac{1}{2}\Delta_k^T \mathcal{H}(M_{k-1})\,\Delta_k,$$

$$\mathcal{H} = \begin{bmatrix} \frac{\partial S^2}{\partial m_1 m_1} & \cdots & \frac{\partial S^2}{\partial m_1 m_{|M|}} \\ \vdots & \ddots & \vdots \\ \frac{\partial S^2}{\partial m_{|M|} m_1} & \cdots & \frac{\partial S^2}{\partial m_{|M|} m_{|M|}} \end{bmatrix}.$$

Lahendame valemi

$$\nabla \widetilde{S} = \mathbf{0}$$

Δ_k suhtes. Saame

$$\mathcal{H}(M_{k-1})\,\Delta_k = -\nabla S(M_{k-1}).$$

See on lineaarvõrrand kujul $A\mathbf{x} = \mathbf{b}$. Rakendame tulemuse:

$$M_k = M_{k-1} + \Delta_k.$$

Levenberg-Marquardt'i meetod I

$$\begin{aligned} S &= \sum_i r_i^2, \\ [\mathcal{H}]_{jk} &= \sum_i 2\left(\frac{\partial r_i}{\partial m_j}\cdot\frac{\partial r_i}{\partial m_k} - r_i\cdot\frac{\partial^2 r_i}{\partial m_j \partial m_k}\right). \end{aligned}$$

Aproksimatsioon miinimumi ligidal:

$$\left[\widetilde{\mathcal{H}}\right]_{jk} = \sum_i 2\frac{\partial r_i}{\partial m_j}\cdot\frac{\partial r_i}{\partial m_k}.$$

Lisaks teeme maatriksi $\widetilde{\mathcal{H}}$ positiivseks skaleerides diagonaalil olevaid elemente:

$$\widetilde{H}^{\lambda} = \widetilde{H} + \lambda\cdot\mathrm{diag}\left(\widetilde{H}\right).$$

See teeb ka leitud sammud väiksemaks ja gradiendi sihis olevaks.

Levenberg-Marquardt'i meetod II

1. Valime $\lambda = 0.0001$.
2. Arvutame
$$\left[\widetilde{\mathcal{H}}\right]_{jk} = \sum_i 2 \frac{\partial r_i}{\partial m_j} \cdot \frac{\partial r_i}{\partial m_k}\bigg|_{M_{k-1}}$$
3. Modifitseerime tema diagonaali:
$$\widetilde{H}^{\lambda} = \widetilde{H} + \lambda \cdot \mathrm{diag}\left(\widetilde{H}\right)$$
4. Kasutades Cholesky maatrikslahutust lahendame
$$\widetilde{H}^{\lambda} \Delta_k = -\nabla S(M_{k-1}) .$$
5. Kui $S(M_{k-1} + \Delta_k) > S(M_{k-1})$, siis $\lambda = 10 \cdot \lambda$ ja kordame sammust 3
6. Võtame
$$\begin{aligned} M_k &= M_{k-1} + \Delta_k , \\ \lambda &= 0.5 \cdot \lambda \end{aligned}$$
ja läheme iteratsioonile $k+1$ sammul 2.

Tulemuse määramatuse hinnang I: K_M

Miinimumi M^* juures $\nabla S(M^*) = \mathbf{0}$ ja Taylori rea kolm esimest liiget:

$$\widetilde{S}(M) \quad = \quad S(M^*) + \frac{1}{2}(M - M^*)^T \widetilde{\mathcal{H}}(M)(M - M^*).$$

Meenutame

$$\begin{aligned} P(M|L) &\simeq C \cdot P(L|M), \\ -\ln P(M|L) &\propto \frac{1}{2}S(M) \simeq \frac{1}{2}\widetilde{S}(M) \propto \frac{1}{2}(M - M^*)^T \frac{1}{2}\widetilde{\mathcal{H}}(M)(M - M^*). \end{aligned}$$

Juhul kui

$$\begin{aligned} M|L &\sim \mathcal{N}(M^*, \mathbf{K}_M), \\ -\ln P(M|L) &\propto \frac{1}{2}(M - M^*)^T \mathbf{K}_M^{-1}(M - M^*). \end{aligned}$$

Siit järeldub:

$$K_M \quad \simeq \quad 2\widetilde{\mathcal{H}}(M^*)^{-1}.$$

Tulemuse määramatuse hinnang II: hii-ruut jaotus

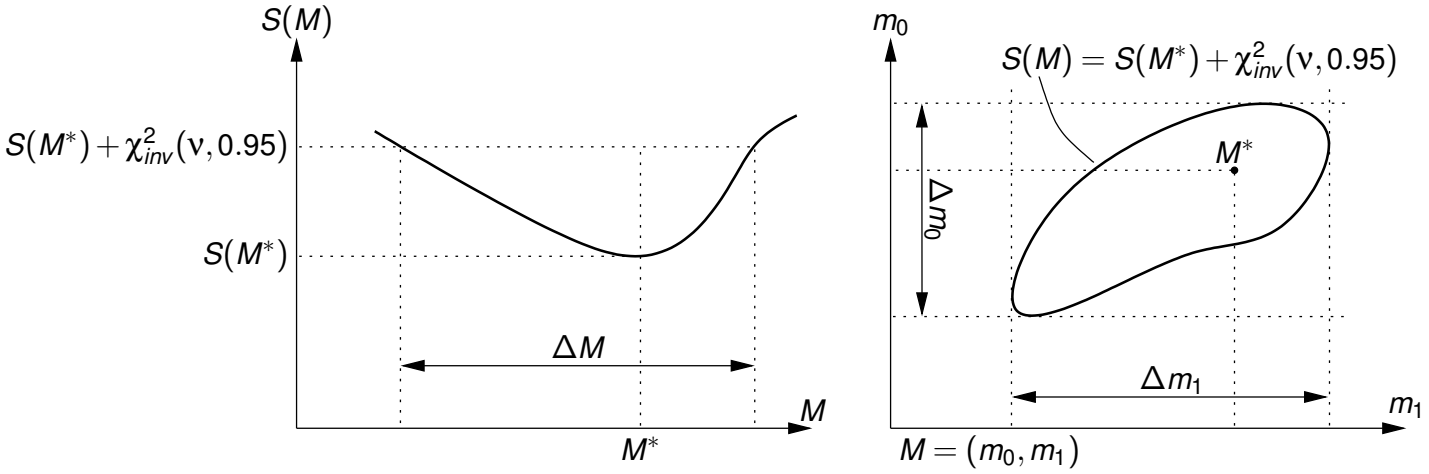
Eeldusel et $r_i \sim \mathcal{N}(0,1)$ on

$$S \sim \chi^2_\nu$$

Vabadusaste ν

$$\text{plokkide arv}(K_L) - |M| \leq \nu \leq |L| - |M|\,.$$

Uurime funktsiooni S kuju miinimumi ümbruses:



Kui $r_i \sim \mathcal{N}(0,1)$ siis

$$\Delta m_i = \pm\sqrt{\chi^2_{inv}(1,p)} \cdot \sqrt{[K_M]_{ii}}.$$

Tulemuse õigsuse hinnang: hii-ruut test

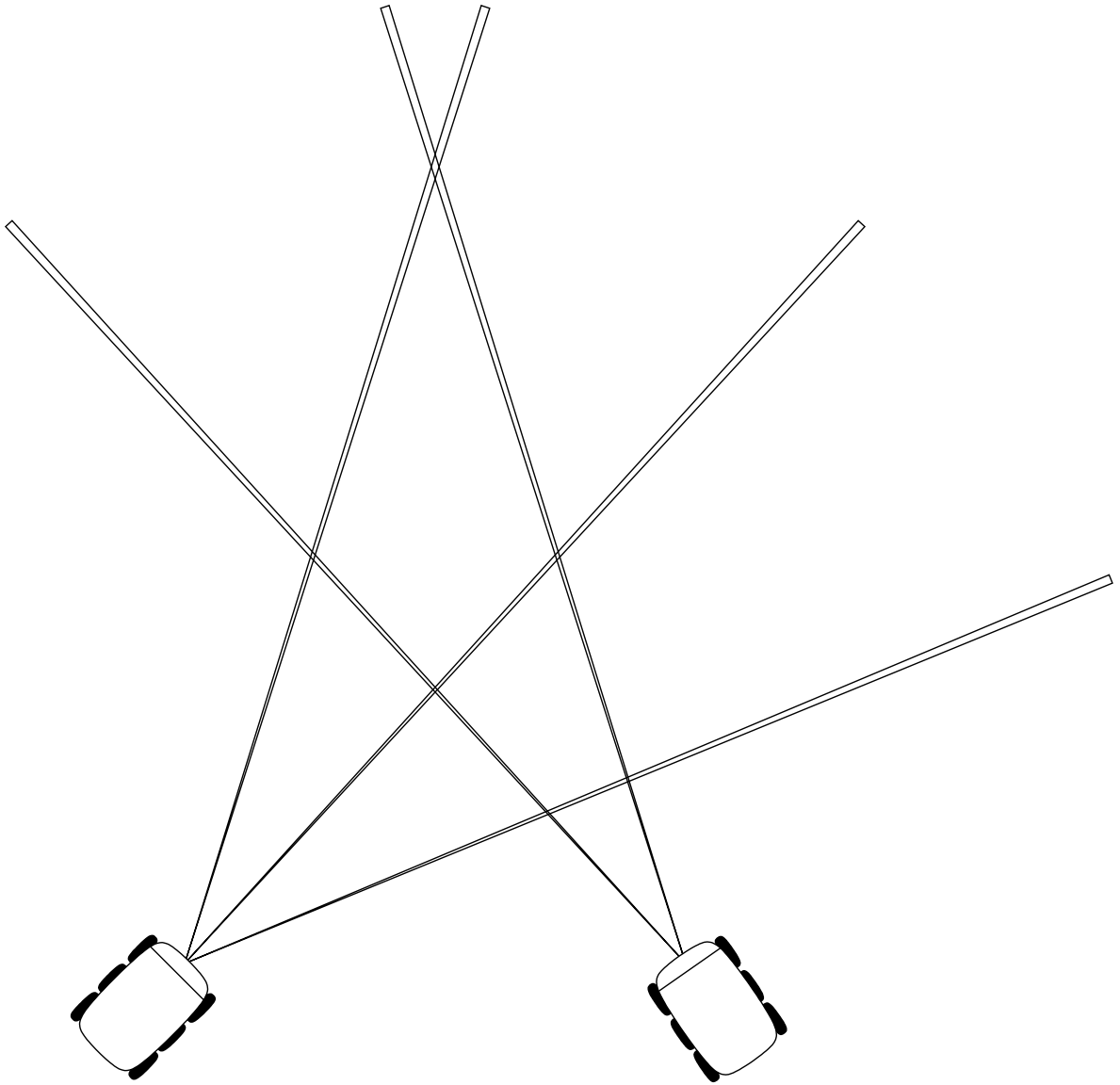
$$S(M^*) < \chi^2_{inv}(\nu, 0.95)\,.$$

Üldiselt:

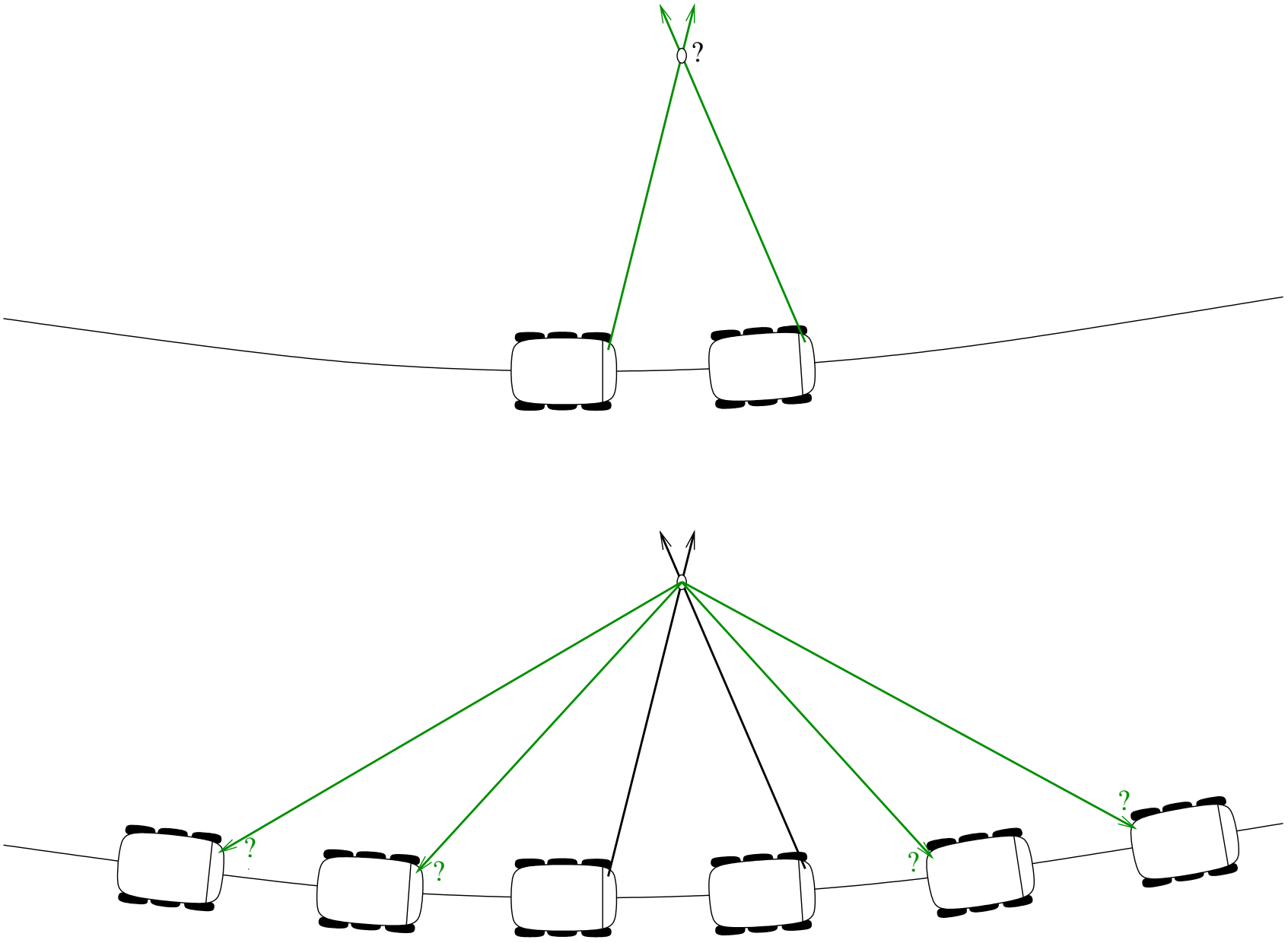
- Kui $S(M^*)/\nu \simeq 1$, on mudel hea
- Kui $S(M^*)/\nu \ll 1$, on L määramatuse hinnangud liiga suured
- Kui $S(M^*)/\nu \gg 1$, on mudel halb

```
>>> from scipy.stats.distributions import chi2
>>> print(chi2.ppf(0.95, df=20))
31.410432844230918
>>> print(chi2.ppf(0.95, df=10))
18.307038053275146
```

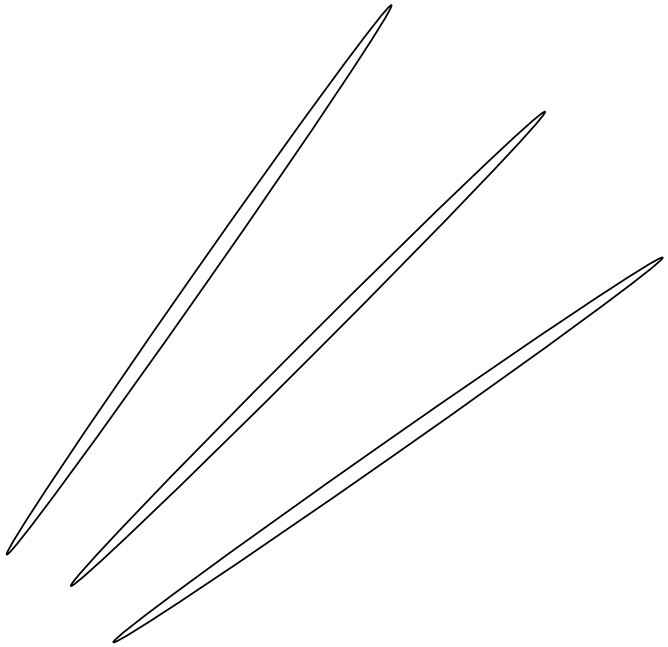
Maamärkide leidmine: vastavusprobleem

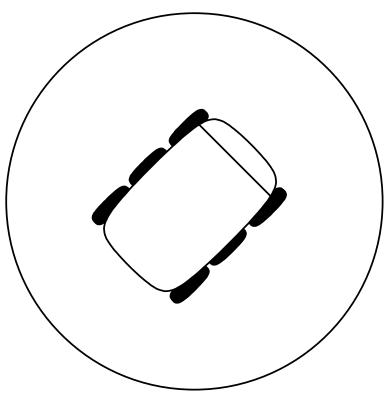


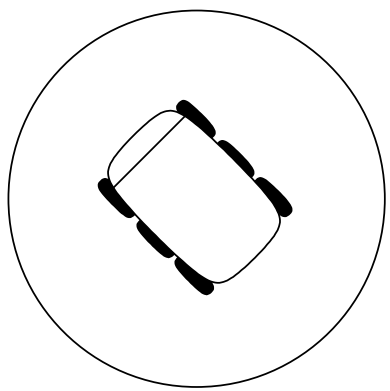
Maamärkide leidmine



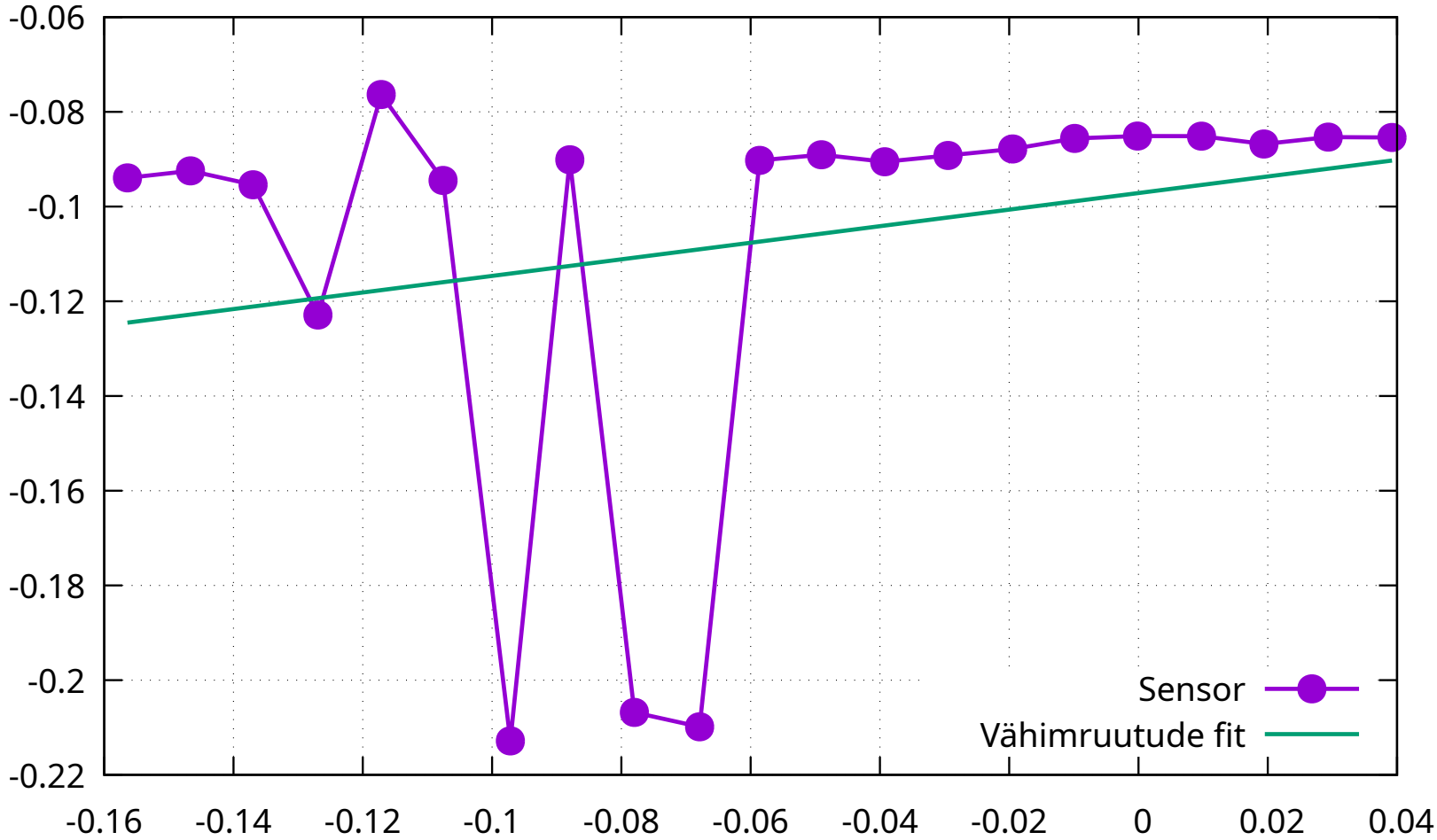
Globaalne kaart vs. lokaalne kaart





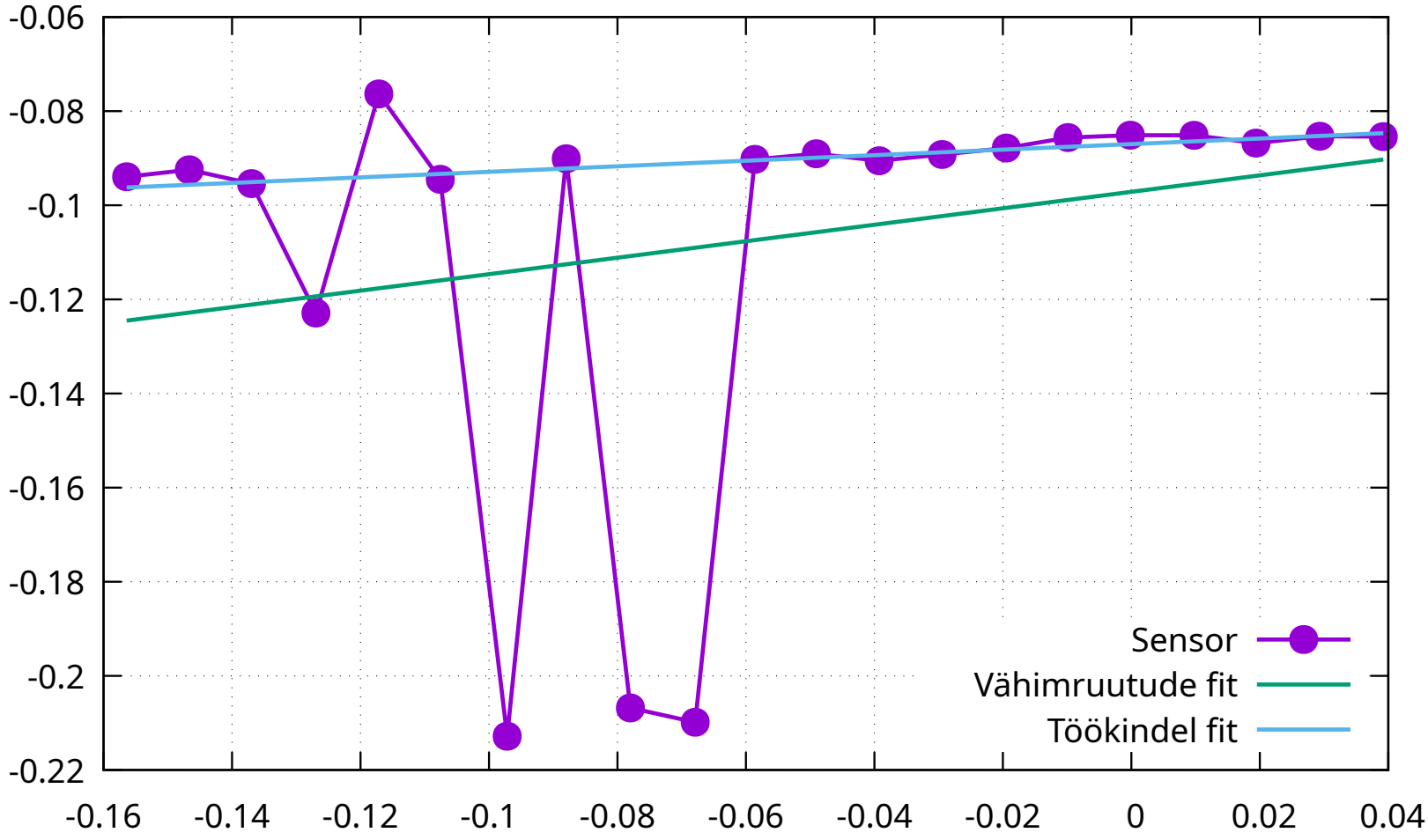


Vastupidavus vigadele I



$$y = ax + b, \quad M = [a, b], \quad S = \sum_i [y_i - (ax_i + b)]^2$$

Vastupidavus vigadele II



$$y = ax + b, \quad M = [a, b], \quad S = \sum_i [y_i - (ax_i + b)]^2$$

RANSAC

- Võtame sensorite lugemist juhusliku alamhulga:

$$\begin{aligned} L_{RNG} &\subset [L_i], \quad |L_{RNG}| < |L| \\ S_{RNG} &= \dots \end{aligned}$$

- Optimeerime S_{RNG} ja leiame miinimumi M_{RNG}
- Loendame kui palju kogu süsteemi $r_i^2(M_{RNG}) < 2.7$. Kui tulemus on parem, võtame selle kui senise parima tulemuse.

$$M_{RNG}^{BEST} = M_{RNG}$$

- Kordame N korda või kui suur enamik $r_i^2 < 2,7$.
- Koostame

$$\begin{aligned} L_{BEST} &\subset [L_i], \quad r_i\left(M_{RNG}^{BEST}\right)^2 < 6.6, \\ S_{BEST} &= \dots \end{aligned}$$

- Optimeerime summat S_{BEST} ja leiame M_{BEST}

Diagonaliseerime S kasutads $K_L = RDR^T$:

$$\begin{aligned} S &= (L-\boldsymbol{\mu}(M))^T K_L^{-1} (L-\boldsymbol{\mu}(M)) \\ &= \left(R^T L - R^T \boldsymbol{\mu}(M)\right) D^{-1} \left(R^T L - R^T \boldsymbol{\mu}(M)\right), \\ &= \sum_i \frac{\left(\left[R^T L\right]_i - \left[R^T \boldsymbol{\mu}(M)\right]_i\right)^2}{\sigma_i^2} = \sum \left(\frac{L_i^D - G_i(M)}{\sigma_i}\right)^2. \\ &= \sum_i \left(r_i^D\right)^2 \end{aligned}$$

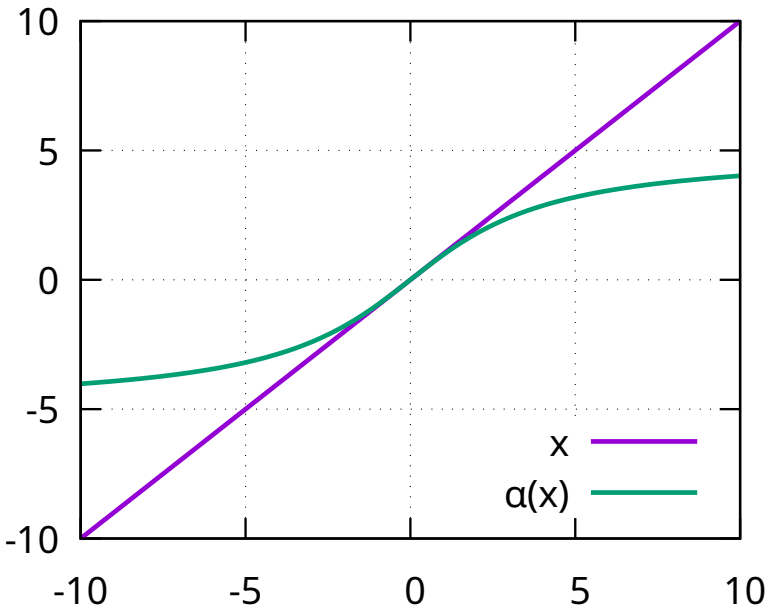
Starship meetod II

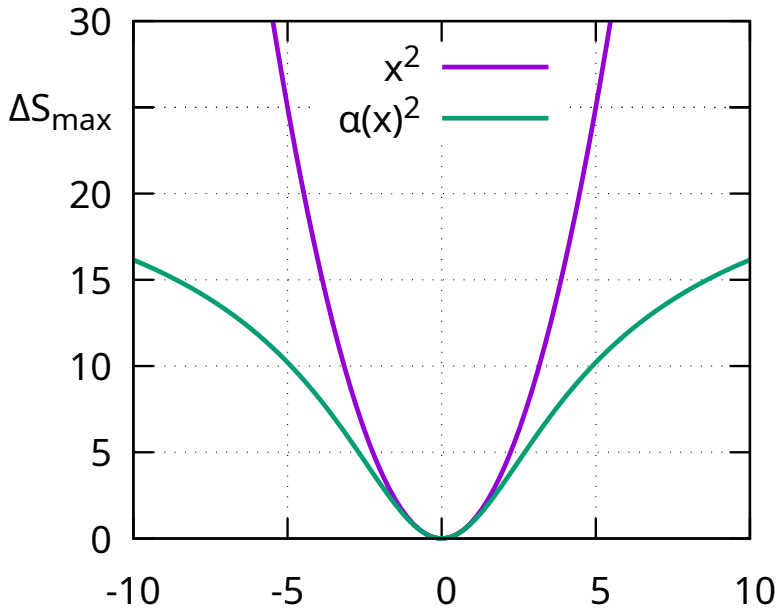
Otsime funktsiooni millega moduleerida jääke r_i^D.

$$\begin{aligned} \alpha(x) &= \gamma \tan^{-1} \frac{x}{\gamma}, \\ \gamma &= \frac{2\sqrt{\Delta S_{max}}}{\pi}. \end{aligned}$$

Meil on

$$\begin{aligned} \alpha(x) &\simeq x, \quad |x| \sim 1, \\ \alpha(x)^2 &\to \Delta S_{max}, \quad x \gg 1. \end{aligned}$$





Starship meetod III

Rakendame funktsiooni $\alpha(x)$ elementidele r_i^D :

$$\begin{aligned} \hat{S} &= \sum_i \alpha\left(r_i^D\right)^2 \\ &= \sum_i \left[\gamma \tan^{-1}\left(\frac{L_i^D - G_i(M)}{\gamma\sigma_i}\right)\right]^2. \end{aligned}$$

Valime

$$\Delta S_{max} \in [10, 25].$$

CAS abil koodi genereerimine I

$$
\begin{aligned}
y &= ax+b, \quad M=[a,b], \quad \sigma_i=\sigma, \\
S &= \sum_i \hat{r}_i^2, \quad \hat{r}_i = \gamma \tan^{-1} \frac{y_i-(ax_y+b)}{\gamma\sigma}.
\end{aligned}
$$

```
1  from sympy import symbols, atan, cse
2  from sympy.printing import ccode
3  import itertools
4  # Describe variables and parameters
5  x,y,a,b,gamma,sigma = symbols('sensor.x sensor.y a b gamma sigma')
6  # Weighted residual
7  plain_r = (a*x+b-y)/sigma
8  # Suppressed residual
9  r = gamma * atan(plain_r / gamma)
10 # First derivative of r against parameters a and b
11 drda = r.diff(a)
12 drdb = r.diff(b)
13 # Extract common subexpressions to optimize and simplify calculation
14 (exprs, results) = cse([r,drda, drdb], optimizations='basic')
15 # Print out the code
16 for key, value in itertools.chain(exprs, zip(('r','drda','drdb'), results)):
17   print('double ' + str(key) + ' = ' + ccode(value, standard='C99') + ';')
18 # print the result assignment code
19 print('G(0) += 2*r*drda;')
20 print('G(1) += 2*r*drdb;')
21 print('H(0,0) += 2*drda*drda;')
22 print('H(0,1) += 2*drda*drdb;')
23 print('H(1,1) += 2*drdb*drdb;')
```

CAS abil koodi genereerimine II

```cpp
#include <Eigen/Core>
#include <vector>
struct LinearModel {
    double sigma = 1, gamma = 2.0 * std::sqrt(25) / M_PI;
    double a = 0, b = 1;
    struct sensor_reading { double x, y; };
    std::vector<sensor_reading> sensor_data;
    void calc_S_gradient_hessian(double& S, Eigen::Vector2d& G, Eigen::Matrix2d& H) {
        S = 0; G = G.Zero(); H = H.Zero();
        for ( auto& sensor : sensor_data ) {
            // Auto-generated code
            double x0 = 1.0/sigma;
            double x1 = a*sensor.x + b - sensor.y;
            double x2 = x0/(1 + pow(x1, 2)/(pow(gamma, 2)*pow(sigma, 2)));
            double r = gamma*atan(x0*x1/gamma);
            double drda = sensor.x*x2;
            double drdb = x2;
            S += r*r;
            G(0) += 2*r*drda;
            G(1) += 2*r*drdb;
            H(0,0) += 2*drda*drda;
            H(0,1) += 2*drda*drdb;
            H(1,1) += 2*drdb*drdb;
        }
        H(1,0) = H(0,1); // H is symmetric
    }
    // ...
};
```

Vaata ka https://github.com/symforce-org/symforce

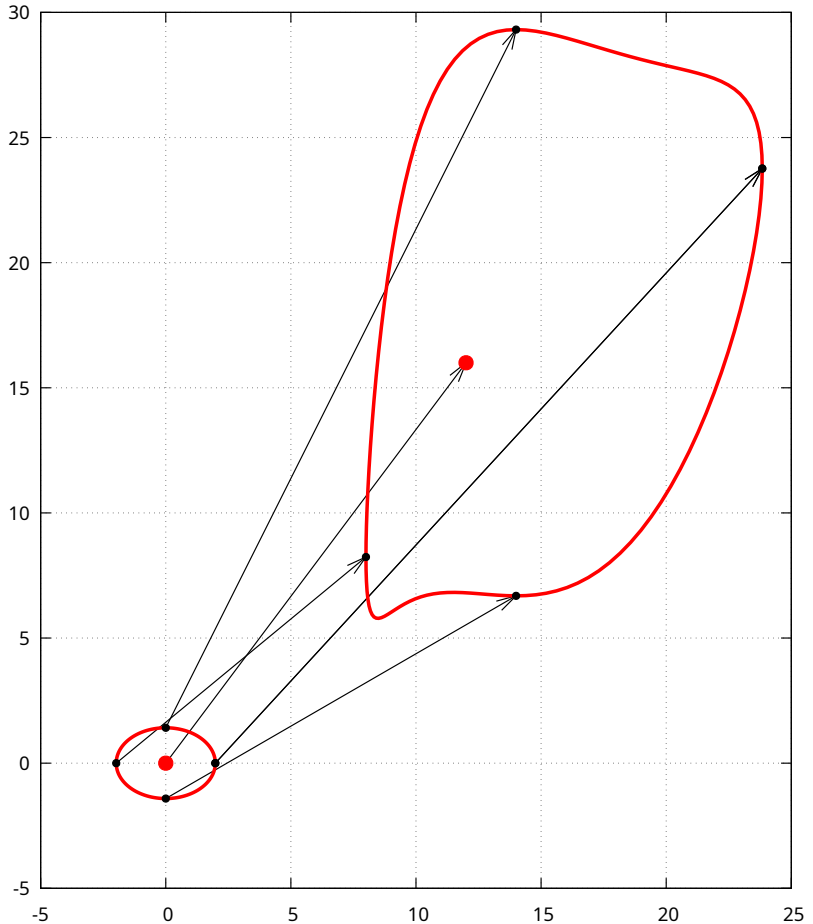
Määramatuse teisendus

Olgu

$$y = f(x \pm \Delta x)\,.$$

Mis on

$$\Delta y = ?.$$



Määramatuse teisendus: lineariseerimine

Olgu

$$\mathbf{y} = f(\mathbf{x}), \qquad \mathbf{x} \sim \mathcal{N}(\mathbf{x}_0, \mathbf{K}_\mathbf{x}).$$

Lineariseerime:

$$\tilde{f}(\mathbf{x}) = f(\mathbf{x}_0) + J_f(\mathbf{x}_0)(\mathbf{x} - \mathbf{x}_0),$$

kus

$$J_f = \begin{pmatrix} \frac{\partial f_1}{\partial x_1} & \cdots & \frac{\partial f_1}{\partial x_{|\mathbf{x}|}} \\ \vdots & \ddots & \vdots \\ \frac{\partial f_{|\mathbf{y}|}}{\partial x_1} & \cdots & \frac{\partial f_{|\mathbf{y}|}}{\partial x_{|\mathbf{x}|}} \end{pmatrix}$$

Siis

$$\begin{aligned} \mathbf{y} &= f(\mathbf{x}_0), \\ \mathbf{K}_\mathbf{y} &= J_f(\mathbf{x}_0) K_\mathbf{x} J_f(\mathbf{x}_0)^T. \end{aligned}$$

“Lõhnata” teisendus (Unscented transform)

Olgu $N = |x|$. Leiame tulemuse ja määramatuse uurides $2N$ punkti $\mathbf{x}$ ümbruses:

$$\begin{aligned} P &= \sqrt{N} \cdot \sqrt{\mathbf{K_x}} \quad | \quad \sqrt{N} R \sqrt{D} \quad \left[\mathbf{K_x} = R\sqrt{D} \left(R\sqrt{D} \right)^T \right], \\ \mathbf{p}_{i\pm} &= f(\mathbf{x} \pm \mathrm{col}_i(P)) \,. \end{aligned}$$

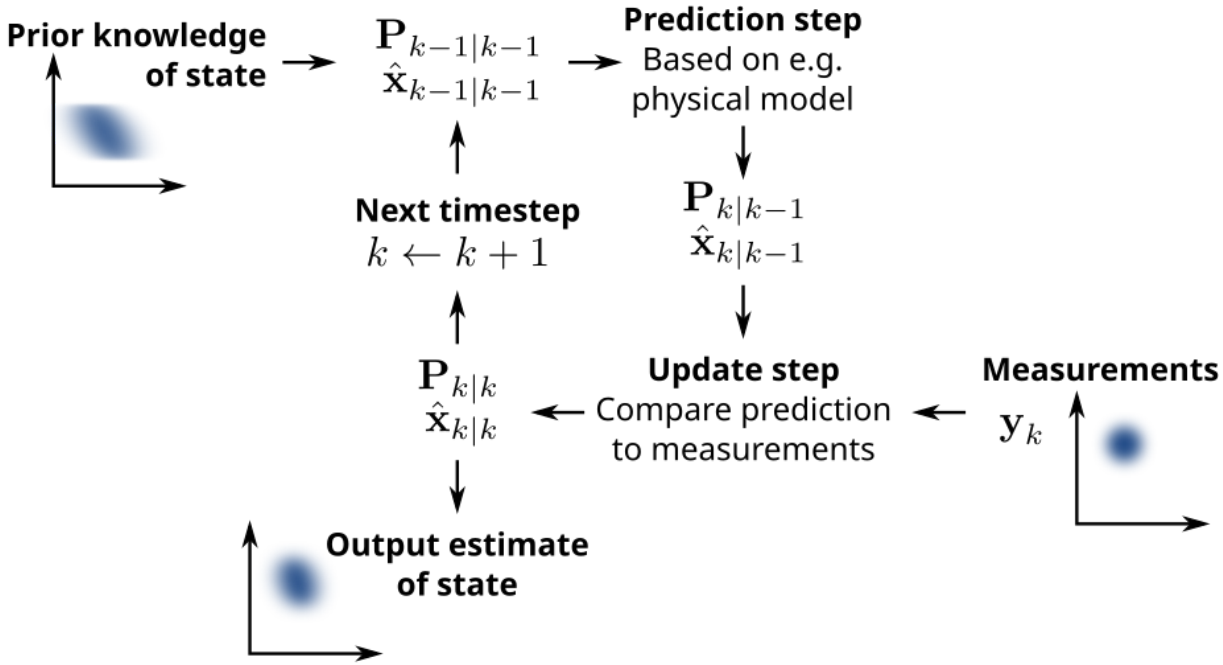
Tulemus on nende keskmine

$$\mathbf{y} = \frac{1}{2N} \sum_{i=1}^{N} \mathbf{p}_{i+} + \mathbf{p}_{i-}$$

ja kovariatsioonimaatriks:

$$\mathbf{K_y} = \frac{1}{2N} \sum_{i=1}^{N} (\mathbf{y} - \mathbf{p}_{i+}) (\mathbf{y} - \mathbf{p}_{i+})^T + (\mathbf{y} - \mathbf{p}_{i-}) (\mathbf{y} - \mathbf{p}_{i-})^T \,.$$

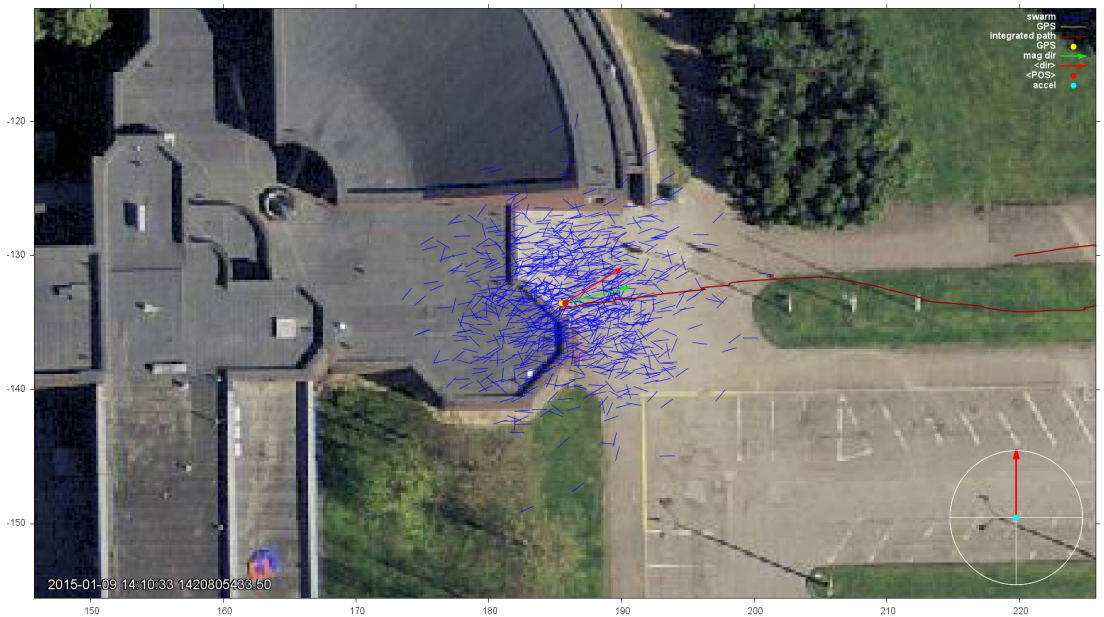
Kalmani filter (Kalman filter)



- lihtsamate mudelite parameetrite leidmine ja jälgimine ajas
  - sensorite kalibratsiooni parameetrite leidmine
- ainult antud hetke olukord, ei hoia ajalugu
- lineariseeritud ülekandemudelid (EKF)
  - lineaarne vähimruutude meetod
- täpsuse suurendamiseks “lõhnata” teisendus

Monte-Carlo meetod: osakeste filter (Particle filter)

- arvutame 1000 roboti hüpoteetilise teekonna
  - robotid liiguvad juhuslikult arvestades odomeetria osa $\mathbf{K}_{L_k}$-s
  - igal robotil on tõenäosus mida uuendame arvestades $\mathbf{K}_{L_k}$-d
- jaotus on esindatud punktipilvega
  - modelleerib ka multimodaalseid jaotusi
- igal sammul
  - kustutame mõned vähem-tõenäolised robotid
  - duplitseerime mõned kõige tõenäolisemad robotid



Sensorite kalibreerimisest

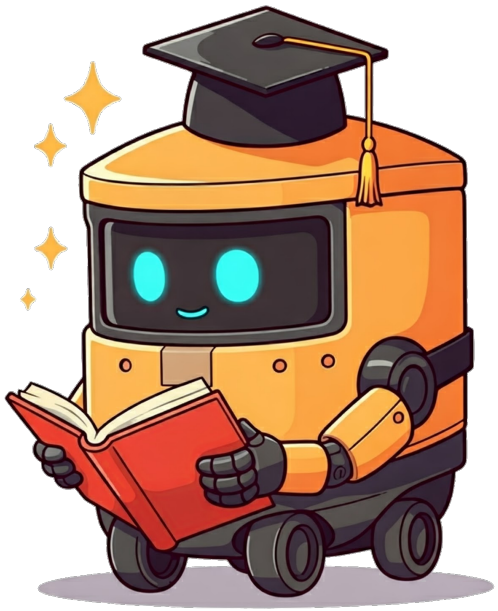
- Kalibreerimata sensorid tekitavad süstemaatilisi vigu
- Süstemaatiliste vigade vastu aitab
  - kalibreerimine
  - nende modelleerimine/tuvastus
- Näiteks ratta diameetrit mõjutab:
  - rehvi kulumine, rehvirõhk, koorem, temperatuur
- Kaamerate asukoha ja orientatsiooni kalibratsioon
  - “creep” – kõik liigub ja paindub aja jooksul
- Kalibreerida tuleb kõike ja pidevalt

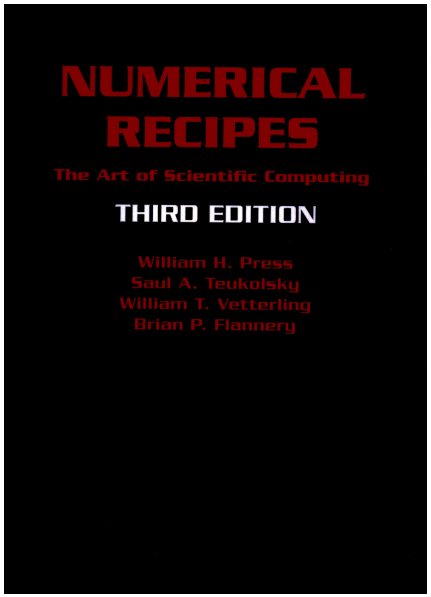
Hõredate maatriksvõrrandite lahendajad

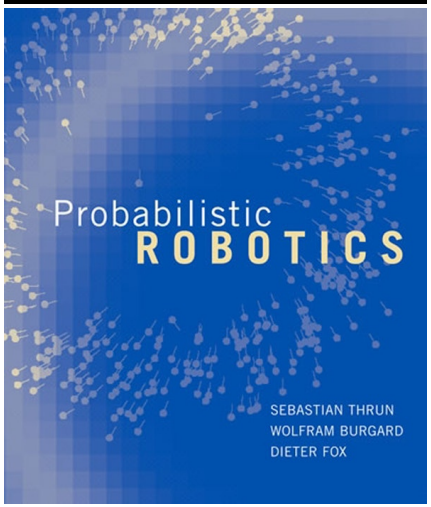
- Eigen @ https://eigen.tuxfamily.org/: `Eigen::SimplicialLLT`
  - `$ sudo apt-get install libeigen3-dev`
- SuiteSparse / CHOLMOD
  - https://github.com/DrTimothyAldenDavis/SuiteSparse
  - `$ sudo apt-get install libcholmod5`
  - Eigen wrapper: `Eigen::CholmodSupernodalLLT`

Raamatud

- Numerical Recipes: The Art of Scientific Computing, William H. Press et al.
- “Probabilistic Robotics” Sebastian Thrun, Wolfram Burgard, Dieter Fox

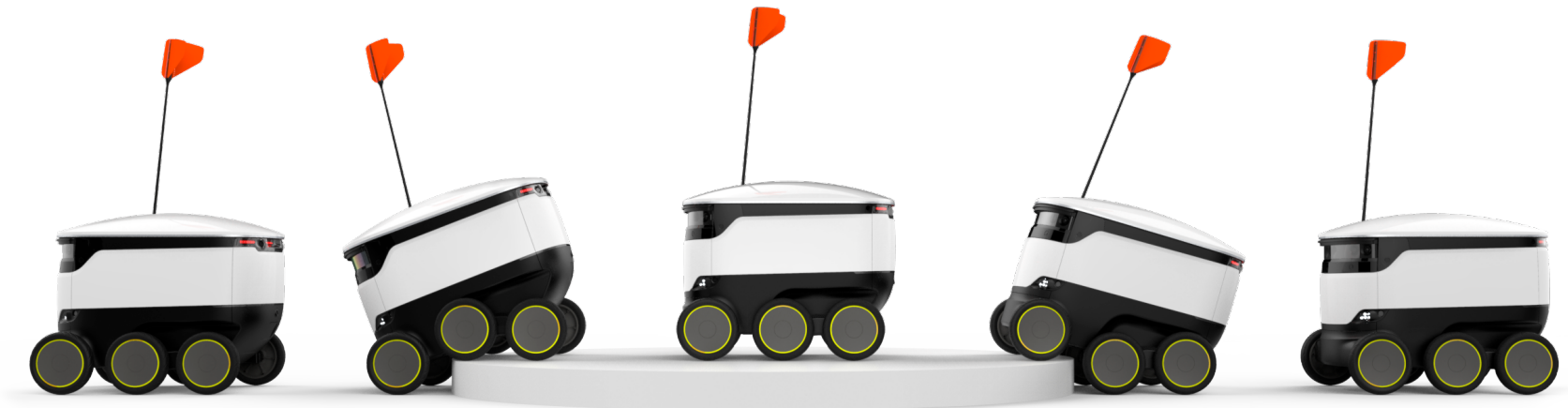




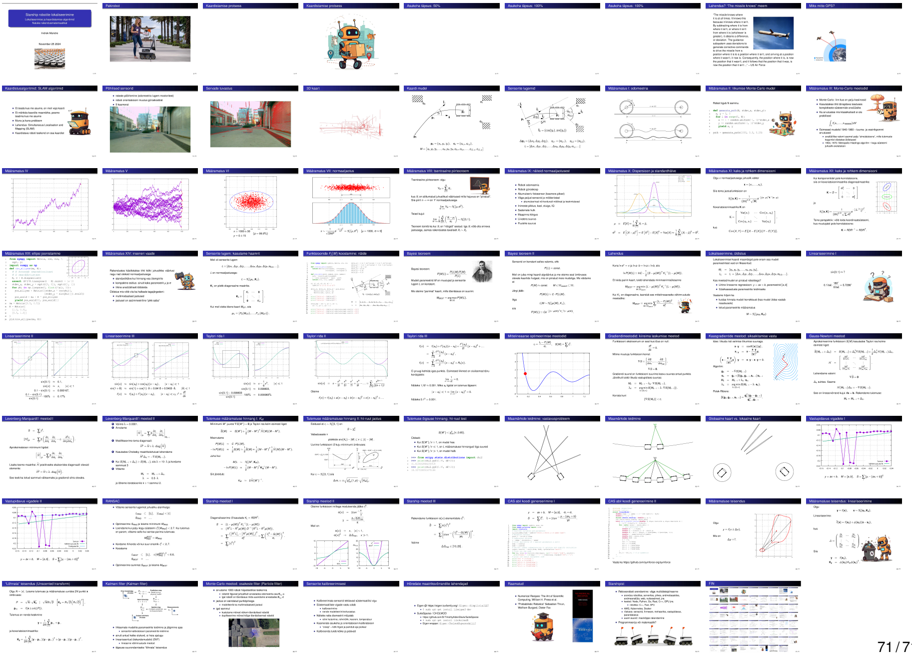


Starshipist

- Pakiveoroboti arendamine: väga multidistsiplinaarne
  - arendus robotites, serverites, pilves, andmebaasides, andmeanalüüs, web, nutiseadmed, jne.
  - keeled: Node, Python, Go, Rust, C++, GPU jne.
    - robotites: C++, Rust, GPU
  - AWS, Kubernetes, Docker
  - riistvara: sensorid, firmware, mehaanika, vastupidavus, parandatavus
  - uuem suund: masinõppe rakendamine
- Programmeerija või matemaatik?



FIN


Starship robotite lokaliseerimine
Indrek Mandre
November 25 2024
Pakirobot
Kaardistamise protsess
Kaardistamise protsess
Asukoha täpsus: 50%
Asukoha täpsus: 100%
Asukoha täpsus: 100%
Lahendus? "The missile knows" meem
Miks mitte GPS?
Kaardistusalgoritmid: SLAM algoritmid
Põhilised sensorid
Servade tuvastus
3D kaart
Kaardi mudel
Sensorite lugemid
Määramatus I: odomeetria
Määramatus II: liikumise Monte-Carlo mudel
Määramatus III: Monte-Carlo meetodid
Määramatus IV
Määramatus V
Määramatus VI
Määramatus VII: normaaljaotus
Määramatus VIII: tsentraalne piirteoreem
Määramatus IX: näiteid normaaljaotusest
Määramatus X: Dispersioon ja standardhälve
Määramatus XI: kaks ja rohkem dimensiooni
Määramatus XII: kaks ja rohkem dimensiooni
Määramatus XIII: ellipsi joonistamine
Määramatus XIV: inseneri vaade
Sensorite lugem: kasutame haamrit
Funktsioonide F_i(M) koostamine: näide
Bayesi teoreem
Bayesi teoreem II
Lahendus
Lokaliseerimine, üldistus
Lineariseerimine I
Lineariseerimine II
Lineariseerimine III
Taylori rida I
Taylori rida II
Taylori rida III
Mittelineaarse optimeerimise meetodid
Gradiendimeetodid: kiireima laskumise meetod
Gauss-Newtoni meetod
Levenberg-Marquardt'i meetod I
Levenberg-Marquardt'i meetod II
Tulemuse määramatuse hinnang I
Tulemuse määramatuse hinnang II: hii-ruut jaotus
Tulemuse õigsuse hinnang: hii-ruut test
Maamärkide leidmine: vastavusprobleem
Maamärkide leidmine
Globaalne kaart vs. lokaalne kaart
Vastupidavus vigadele I
Vastupidavus vigadele II
RANSAC
Starship meetod I
Starship meetod II
Starship meetod III
CAS abil koodi genereerimine I
CAS abil koodi genereerimine II
Määramatuse teisendus
Määramatuse teisendus: lineariseerimine
"Lõhnata" teisendus (Unscented transform)
Kalmani filter (Kalman filter)
Monte-Carlo meetod: osakeste filter (Particle filter)
Sensorite kalibreerimisest
Hõredate maatriksvõrrandite lahendajad
Raamatud
Starshipist
FIN