$$\begin{pmatrix} \Delta z \\ \Delta y \end{pmatrix} = \begin{pmatrix} a \\ b \end{pmatrix} + \begin{pmatrix} Z & L \\ J & Y \end{pmatrix} \times \begin{pmatrix} \Delta x \\ |\Delta z| \end{pmatrix}$$

- Evaluate abs normal form:
  - Geg: $a, b, Z, L, J, Y, \Delta x$
  - Ges: $\Delta z, \Delta y$
- Calculate Gradient
  - Geg: $a, b, Z, L, J, Y, \Delta Z$
  - Ges: Gradient γ, Γ
- Solve abs-normal form system
  - Geg: $a, b, Z, L, J, Y, \Delta y$
  - Ges: $\Delta x, \Delta Z$

Programmiersprachen

- Python 3.5: Prototyping und Serial Performance benchmarks
- Cuda C++: Implementierung der paralleln ABSNF Aufgaben

Annahmen:

1. Global memory der GPU ist groß genug um alle benötigten Datenstrukturen zeitgleich zu halten
2. Daten werden vektorisiert übergeben
3. Benutzen Lineare Algebra so weit wie möglich
4. Sofern möglich mappe alle Problem auf existierende Librariers

Programmiersprachen

- Python 3.5: Prototyping und Serial Performance benchmarks
- Cuda C++: Implementierung der paralleln ABSNF Aufgaben

Annahmen:

1. Global memory der GPU ist groß genug um alle benötigten Datenstrukturen zeitgleich zu halten
2. Daten werden vektorisiert übergeben
3. Benutzen Lineare Algebra so weit wie möglich
4. Sofern möglich mappe alle Problem auf existierende Librariers

- utils.hpp test_utils.hpp
- absnf.h test_absnf.cu
- cuutils.h test_cuutils.cu
- tabsnf.h test_tabsnf.h
- make
- absnf.py

Benutzte Libraries:

- cuBLAS (cuda Basic Linear Algebra Subprograms)
  - Matrix Vector operations
  - Matrix Matrix operations
- cuSOLVER
  - Matrix factorization
  - Triangular solve
- C++ STL

$$\begin{pmatrix} \Delta z_1 \\ \Delta z_2 \\ \Delta z_3 \\ \Delta z_4 \end{pmatrix} = \begin{pmatrix} k_1 \\ k_2 \\ k_3 \\ k_4 \end{pmatrix} + \begin{pmatrix} 0 & 0 & 0 & 0 \\ L_{2,1} & 0 & 0 & 0 \\ L_{3,1} & L_{3,2} & 0 & 0 \\ L_{4,1} & L_{4,2} & L_{4,3} & 0 \end{pmatrix} \times \begin{pmatrix} |\Delta z_1| \\ |\Delta z_2| \\ |\Delta z_3| \\ |\Delta z_4| \end{pmatrix}$$

$$k = a + Z \times \Delta x$$

$$\begin{aligned}
\Delta z_1 &= \underbrace{L_1 \times |\Delta z|}_{=0} + k_1 = k_1 \\
\Delta z_2 &= L_2 \times |\Delta z| + k_2 \\
&= L_{2,1} \times |\Delta z_1| + k_2 \\
\Delta z_3 &= L_3 \times |\Delta z| + k_3 \\
&= L_{3,1} \times |\Delta z_1| + L_{3,2} \times |\Delta z_2| + k_3 \\
\Delta z_4 &= L_4 \times |\Delta z| + k_4 \\
&= L_{4,1} \times |\Delta z_1| + L_{4,2} \times |\Delta z_2| + L_{4,3} \times |\Delta z_3| + k_4
\end{aligned}$$

Gegeben:

$$a, b, Z, L, J, Y, m, n, s, \Delta Z$$

Gesucht:

$$\gamma, \Gamma$$

$$\gamma = b + Y\Sigma(I - L\Sigma)^{-1}a$$
$$\Gamma = J + Y\Sigma(I - L\Sigma)^{-1}Z$$

$$\Sigma = Diag(Sign(\Delta z))$$

$$\begin{pmatrix} \Delta z \\ \Delta y \end{pmatrix} = \begin{pmatrix} a \\ b \end{pmatrix} + \begin{pmatrix} Z & L \\ J & Y \end{pmatrix} \times \begin{pmatrix} \Delta x \\ |\Delta z| \end{pmatrix}$$

$$\begin{pmatrix} \times & \times & \times & \times \\ 0 & \times & \times & \times \\ 0 & 0 & \times & \times \\ 0 & 0 & 0 & \times \\ a & b & c & d \end{pmatrix}$$

$$\begin{pmatrix} \Delta z \\ \Delta y \end{pmatrix} = \begin{pmatrix} a \\ b \end{pmatrix} + \begin{pmatrix} Z & L \\ J & Y \end{pmatrix} \times \begin{pmatrix} \Delta x \\ |\Delta z| \end{pmatrix}$$

$$\begin{pmatrix} \Delta z_1 \\ \vdots \\ \Delta z_s \\ \Delta y_1 \\ \vdots \\ \Delta y_m \end{pmatrix} = \begin{pmatrix} a_1 \\ \vdots \\ a_s \\ b_1 \\ \vdots \\ b_m \end{pmatrix} + \begin{pmatrix} Z_{1,1} & \dots & Z_{1,n} & 0 & \dots & 0 \\ \vdots & \ddots & \vdots & L_x & \ddots & 0 \\ Z_{s,1} & \dots & Z_{s,n} & L_x & L_x & 0 \\ J_{1,1} & \dots & J_{1,n} & Y_{1,1} & \dots & Y_{1,s} \\ \vdots & \ddots & \vdots & \vdots & \ddots & \vdots \\ J_{m,1} & \dots & J_{m,n} & Y_{m,1} & \dots & Y_{m,s} \end{pmatrix} \times \begin{pmatrix} \Delta x_1 \\ \vdots \\ \Delta x_n \\ |\Delta z_1| \\ \vdots \\ |\Delta z_s| \end{pmatrix}$$

So wenig wie möglich selbst machen
CUBLAS, CUSOLVE, THRUST, C++ STL
Genpgend Speicher vorhanden