Gegeben:

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

Gesucht:

$$\gamma, \Gamma$$

Wobei:

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

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

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

Brauchen:

$$\Sigma(I - L\Sigma)^{-1}$$

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

Fallstricken:

- Sparse Matrix Σ
- Inverse $(I - L\Sigma)^{-1}$

Sei:

$$\Delta z = [-3, 0, 4, -1]$$

Dann gilt für $I - L\Sigma$:

$$I - \begin{pmatrix} 0 & 0 & 0 & 0 \\ L_{2,1} & 0 & 0 & 0 \\ L_{3,1} & L_{3,2} & 0 & 0 \\ L_{4,1} & L_{4,2} & L_{4,3} & 0 \end{pmatrix} \times \begin{pmatrix} -1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & -1 \end{pmatrix} = \begin{pmatrix} 1 & 0 & 0 & 0 \\ -L_{2,1} & 1 & 0 & 0 \\ -L_{3,1} & 0 & 1 & 0 \\ -L_{4,1} & 0 & 0 & 1 \end{pmatrix}$$

das entspricht folgenden Operationen:

- Hinzufügen einer Hauptdiagonalen
- Skalieren der Spalten von L mit den Vorzeichen von Δz

Besser dieses als Operation zu implementieren. Das Auflösen der unteren Dreiecksmatrix $(I - L\Sigma)^{-1}$ übernimmt CUBLAS.

Aufgabe 1) Gradient

Implementierung

```
1   template <typename T>
2   void gradient(T *a, T *b,
3           T *Z, T *L,
4           T *J, T *Y,
5           T *dz,
6           T *Tss, T *I, T *K,
7           int m, int n, int s,
8           int gridsize, int blocksize,
9           T *gamma, T *Gamma)
10    //  d_Tss = diag(1) - L * diag(sign(dz))
11    initTss <<<gridsize, blocksize >>>(d_Tss,d_L, d_dz, s, s*s);
12    //  d_I = diag(1) // room for improvement, operations can be merged
13    initIdentity <<<gridsize, blocksize >>> (d_I, s);
14    //  d_I = d_Tss * X
15    getTriangularInverse(handle, d_Tss, d_I, s);
16    //  d_I = d_I * diag(sign(dz))
17    multWithDz <<<gridsize, blocksize >>>(d_I, d_dz, s);
18    //  d_K = d_Y * d_I
19    cublasDgemm(.,d_Y,.,d_I,d_K,));
20    //  d_gamma = d_b
21    //  d_Gamma = J
22    cudaMemcpy(d_gamma, d_b,.);
23    cudaMemcpy(d_Gamma, d_J,.);
24    //  d_gamma = d_gamma + K*a
25    cublasDgemv(.,d_K,., d_a,., d_gamma,.);
26    //  d_Gamma = d_Gamma + K*Z
27    cublasDgemm(.,d_K,d_Z,d_Gamma,m));
28  }
```

Speicherkomplexität:

a	b	Z	L	J	Y	Δz	γ	Γ	Tss	I	K
s	m	$s * n$	$s * s$	$m * n$	$m * s$	s	m	$m * n$	$s * s$	$s * s$	$m * s$

Bei $m = n = s$:

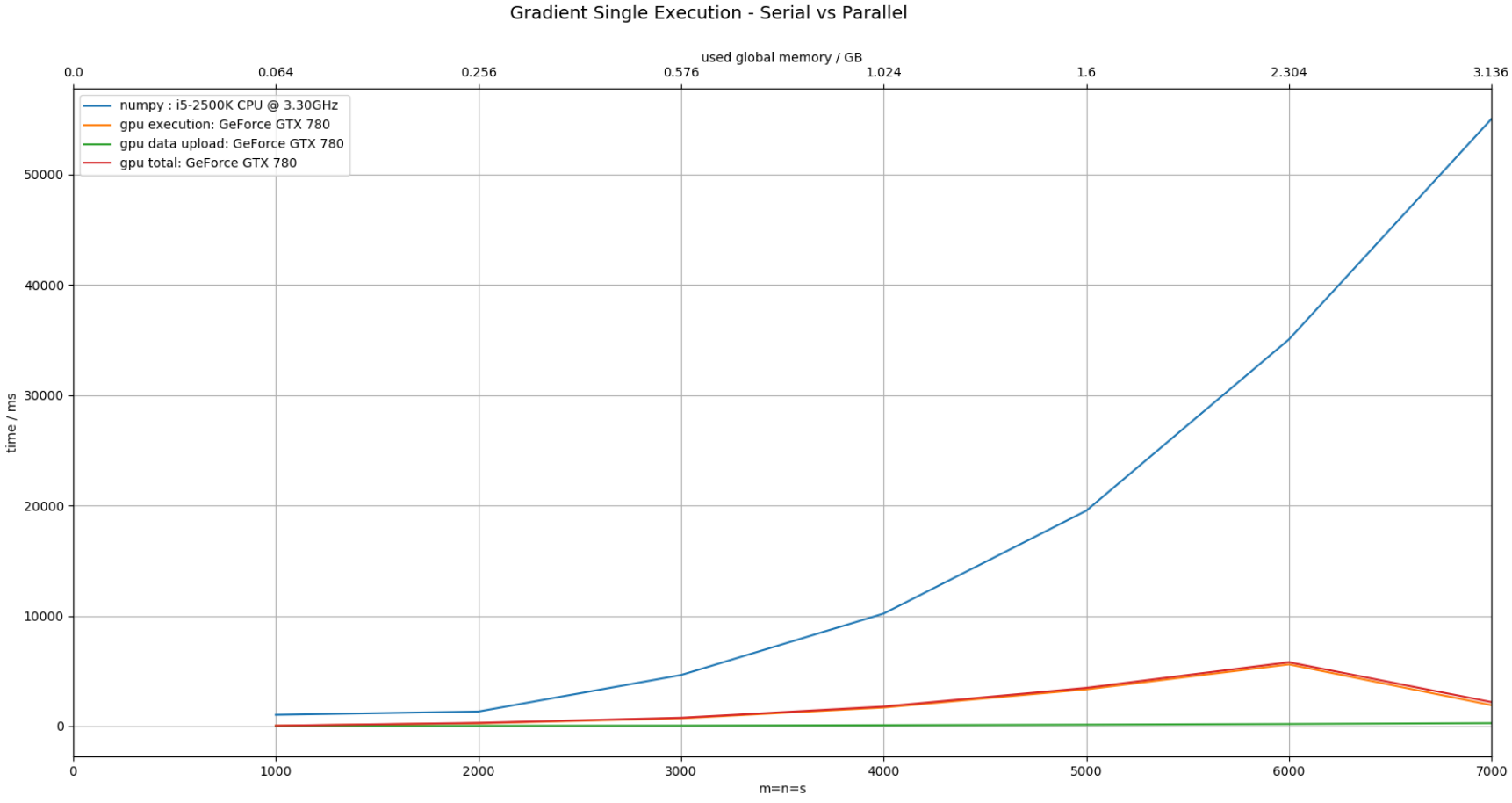
$$8s^2 + 4s \times sizeof(type)$$

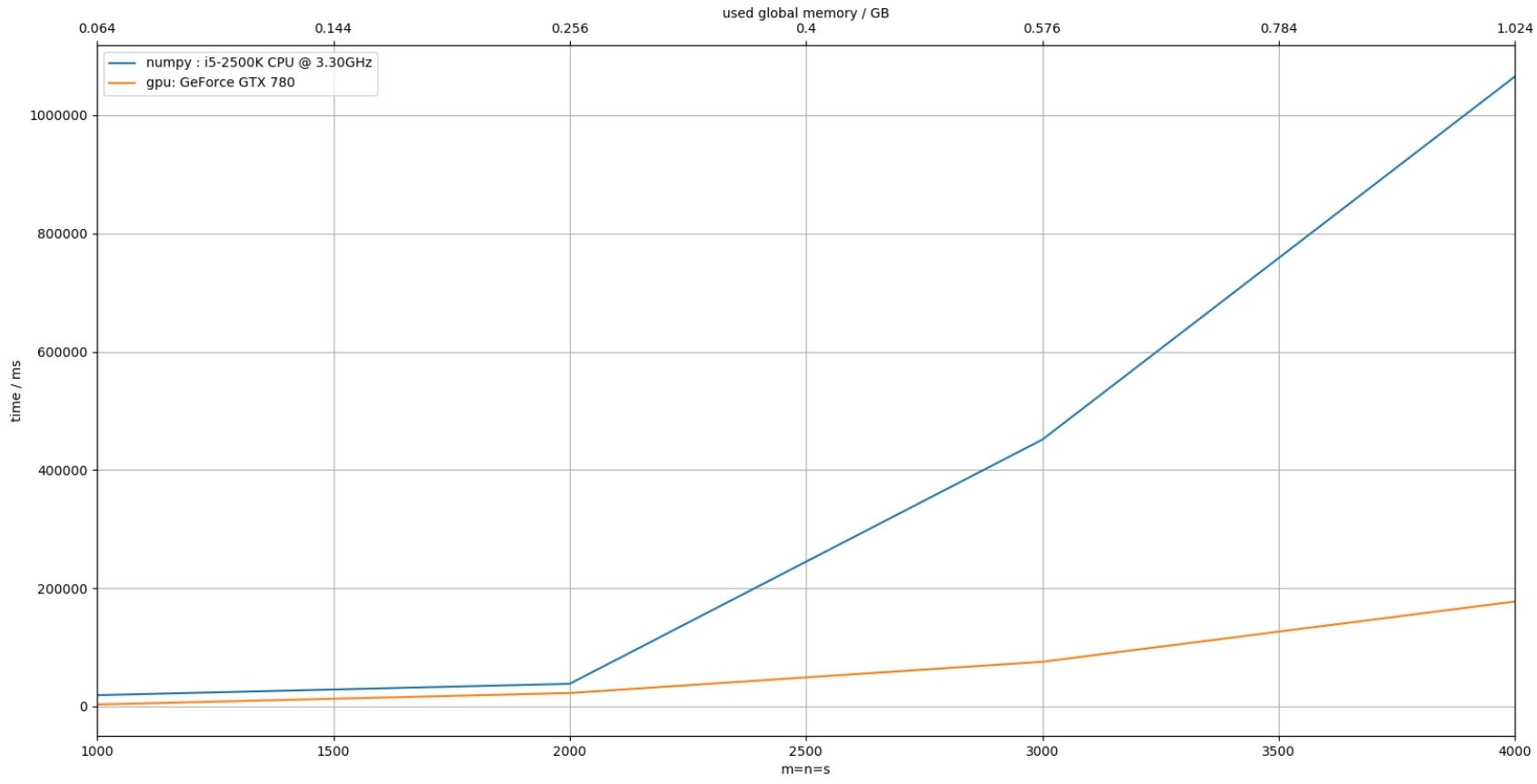
- $m = n = s = 1000 : 0.064$ GB
- $m = n = s = 5000 : 1.6$ GB
- $m = n = s = 10.000 : 6.40$ GB

Komplexität ($m = n = s$):

Funktion	Komplexität Seriell
initTss()	s^2
initIdentity()	s^2
getTriangularInverse()	s^2 (backsubstitution)
multWithDz()	s^2
cublasDgemm()	s^3
cublasDgemv()	s^2
cudaMemcpy()	s

Lässt sich alles gut parallel ausführen !!!


Gradient Single Execution - Serial vs Parallel
used global memory / GB
0.0
0.064
0.256
0.576
1.024
1.6
2.304
3.136
numpy : i5-2500K CPU @ 3.30GHz
gpu execution: GeForce GTX 780
gpu data upload: GeForce GTX 780
gpu total: GeForce GTX 780
time / ms
50000
40000
30000
20000
10000
0
0
1000
2000
3000
4000
5000
6000
7000
m=n=s


Gradient 100 Executions - Serial vs Parallel
used global memory / GB
0.064
0.144
0.256
0.4
0.576
0.784
1.024
numpy : i5-2500K CPU @ 3.30GHz
gpu: GeForce GTX 780
1000000
800000
600000
400000
200000
0
time / ms
1000
1500
2000
2500
3000
3500
4000
m=n=s

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

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

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

BLOCKSIZE, GRIDSIZE OPTIMIZATION
ROW FORMAT, COL FORMAT
INTERFACE ???