Parallel Reduction to Condensed Forms for Symmetric Eigenvalue Problems using Fine-Grained and Memory-Aware Kernels

Azzam Haidar

INNOVATIVE COMPUTING LABORATORY
THE UNIVERSITY of TENNESSEE

Outline

1. Part I : overview
2. a story
3. Part I : The description
4. Part I : The results
5. Part II : advertisement
6. The Burns supper @ cerfacs: some pics
7. I appreciate your contribution and Thanks

ICL UT

General Overview: the Eigenproblem algorithms

Two-stages approach:

1. Symmetric EVP
    - Tri-Diagonalization Reduction + solve.

2. Singular Value Decomposition
    - Bi-Diagonalization Reduction + solve.

General Overview: the Eigenproblem algorithms

First stage require:
- 90% if only eigenvalues
- 50% if eigenvalues and eigenvectors

Two-stages approach:

1. Symmetric EVP
   - Tri-Diagonalization Reduction + solve.
2. Singular Value Decomposition
   - Bi-Diagonalization Reduction + solve.

Mathematical story

Episode 1/∞ : last century

Miss
$AX = \lambda X$

Mr
Scientific

Mathematical story

Episode 2/∞ : scientific challenge

Miss
$AX = \lambda X$

Big challenge: How to win the heart of Miss ($AX=\lambda X$)

Goal

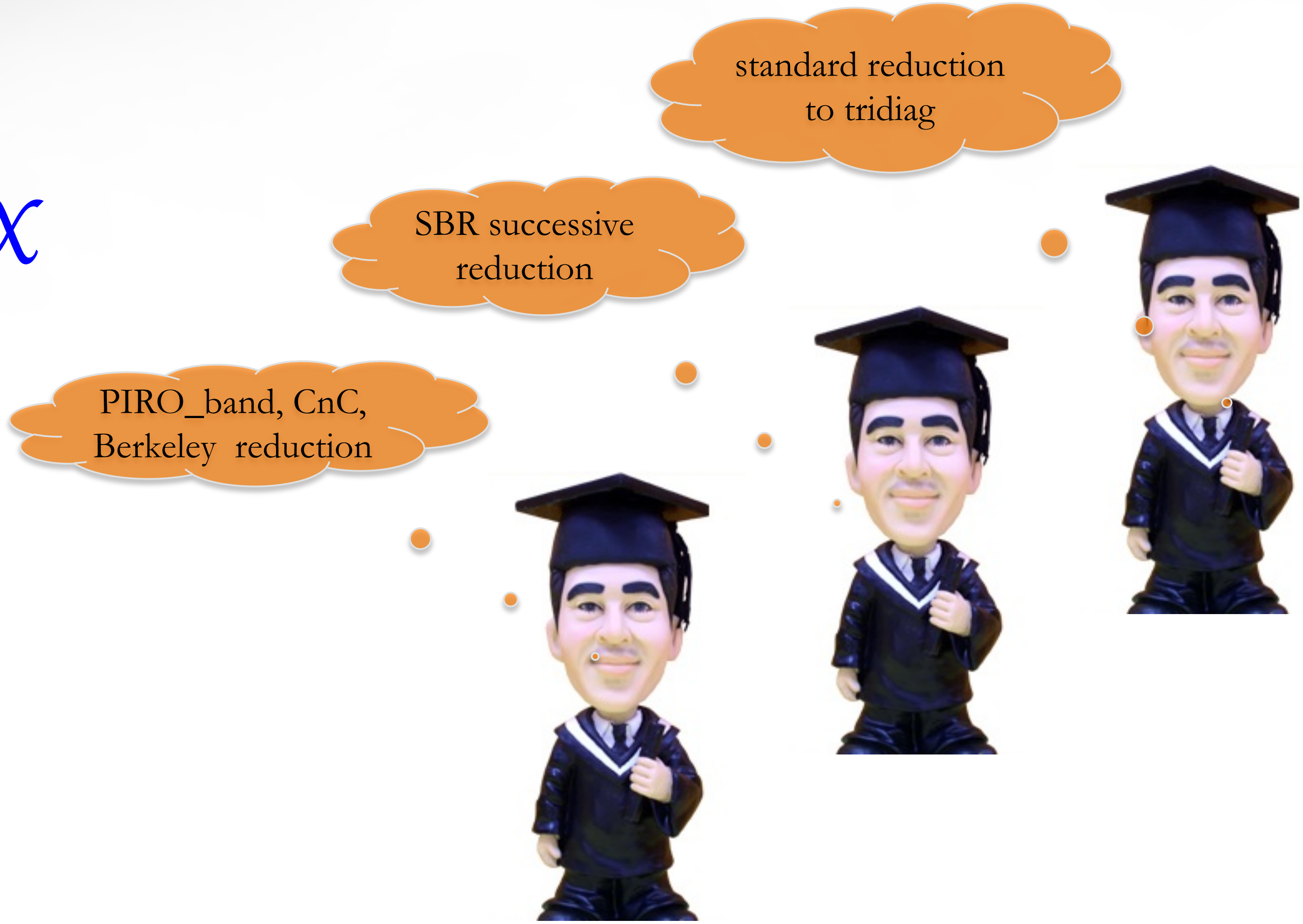
Mr
Scientific

Mathematical story

Episode 1/∞ : propositions

Miss

$AX = \lambda X$



Mathematical story

Episode 1/∞ : my view

for you:
I haven't got a clue
but let me start with a brief overview

Miss
$AX = \lambda X$

Mathematical story

Episode 1/∞ : my view

Miss

$$AX = \lambda X$$

for you:
I haven't got a clue
but let me start with a brief overview

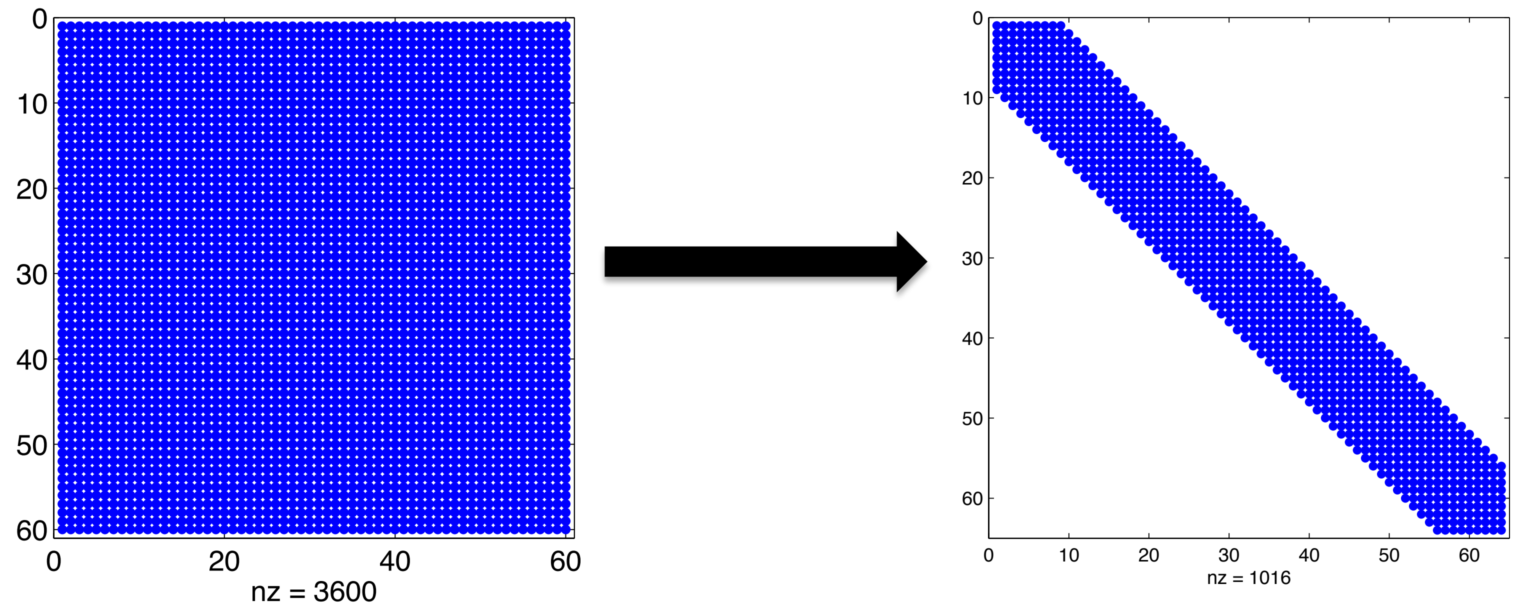
for her:
I haven't got a clue
but let me start by saying I love you

Outline

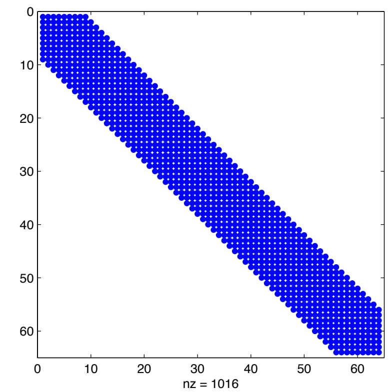
1. Part I : overview
2. a story
3. Part I : The description
4. Part I : The results
5. Part II : advertisement
6. The Burns supper @ cerfacs: some pics
7. I appreciate your contribution and Thanks

Description: the reduction algorithms

- Standard algorithm are very slow
- We propose to first reduce the dense matrix to band. This technique give us a very good performance.



Observations



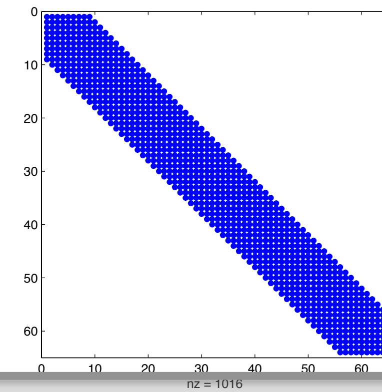
- **Reduction achieved ONLY to band forms.**
- **Need to go to FULL reduction ?**

1. **Investigate research on band solvers:**
   - Re-think about techniques such as Band Divide and Conquer,
   - Open question…

2. **Bulge Chasing:**
   - Relies on Blas-1 operations,
   - Most of the existing techniques consists on sequential operations,
   - Expensive, memory bound and lack of efficiency.

Observations

- Reduction achieved **ONLY** to band forms.
- Need to go to **FULL** reduction ?

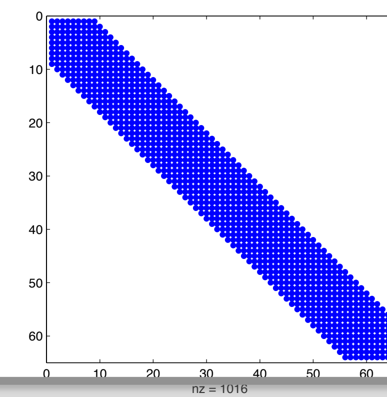


1. **Investigate research on band solvers:**
   - Re-think about techniques such as Band Divide and Conquer,
   - Open question…

2. **Bulge Chasing:**
   - Relies on Blas-1 operations,
   - Most of the existing techniques consists on sequential operations,
   - Expensive, memory bound and lack of efficiency.

Observations

- Reduction achieved ONLY to band forms.
- Need to go to FULL reduction ?

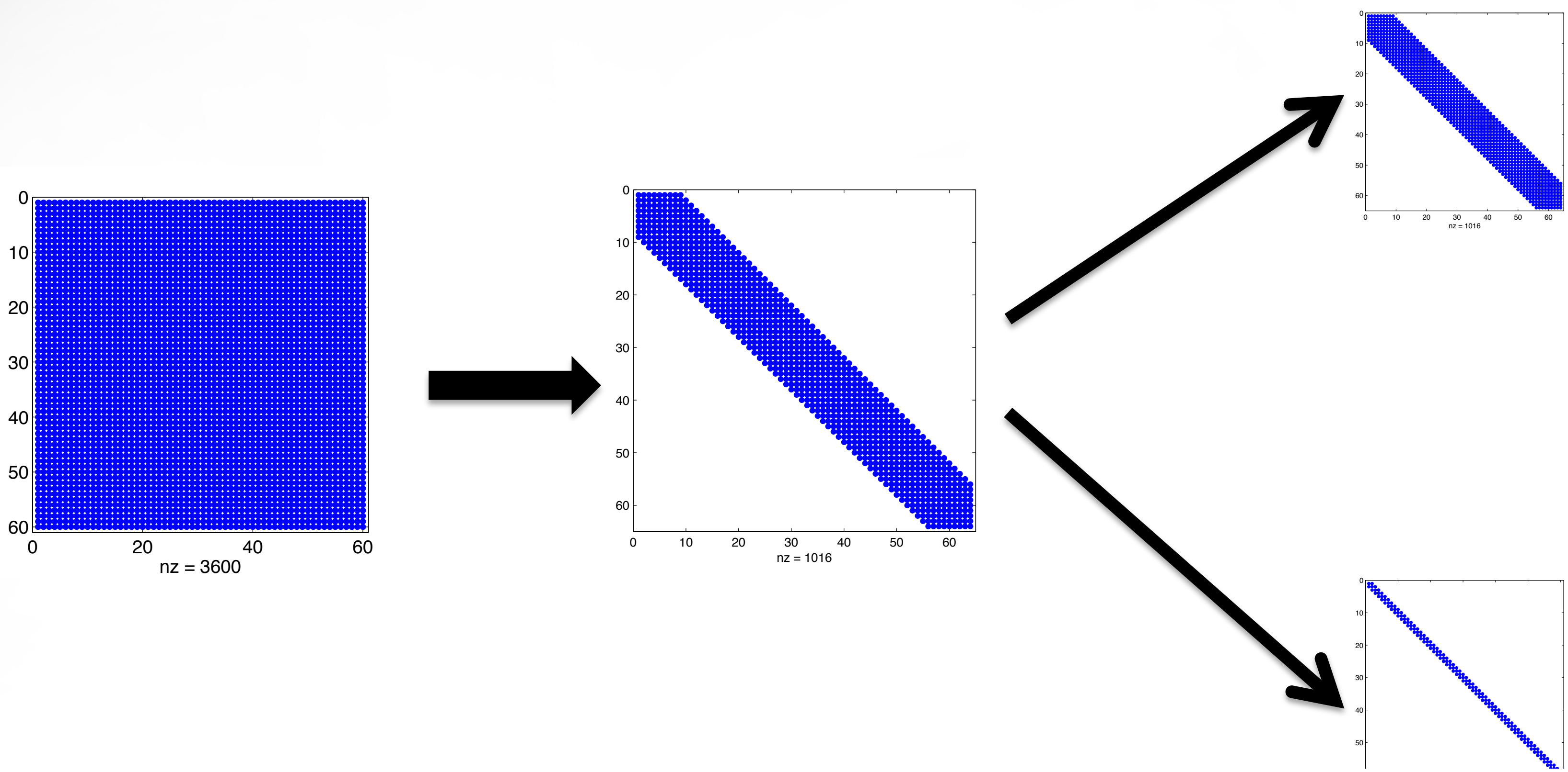


1. **Investigate research on band solvers:**
   - Re-think about techniques such as Band Divide and Conquer,
   - Open question…

2. **Bulge Chasing:**
   - Relies on Blas-1 operations,
   - Most of the existing techniques consists on sequential operations,
   - Expensive, memory bound and lack of efficiency.

Observations

1- Divide and conquer on Band



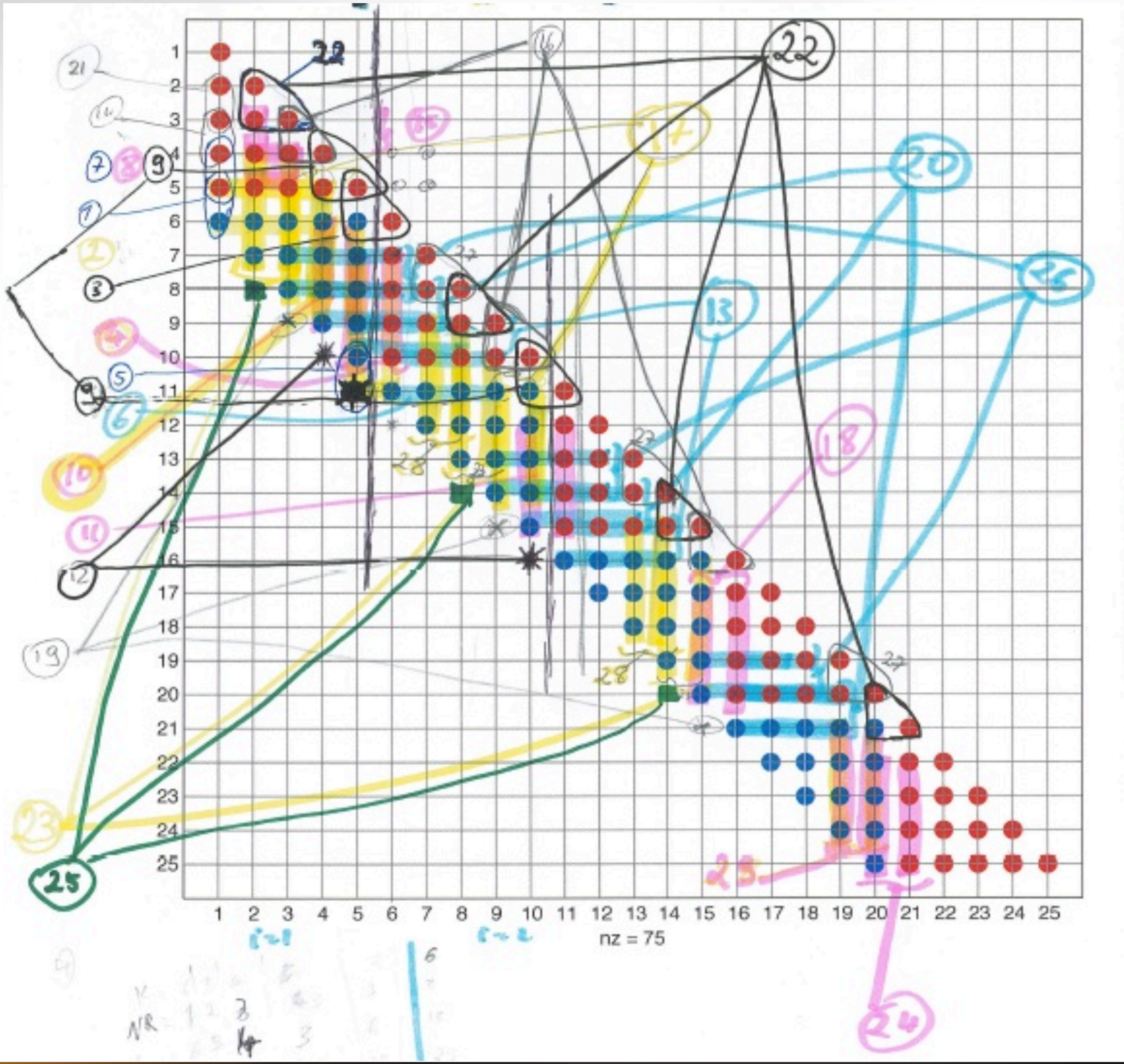
2- bulge chasing then D&C on tridiag

Description: the reduction algorithms

step -2-

The **bulge chasing** of the band matrix

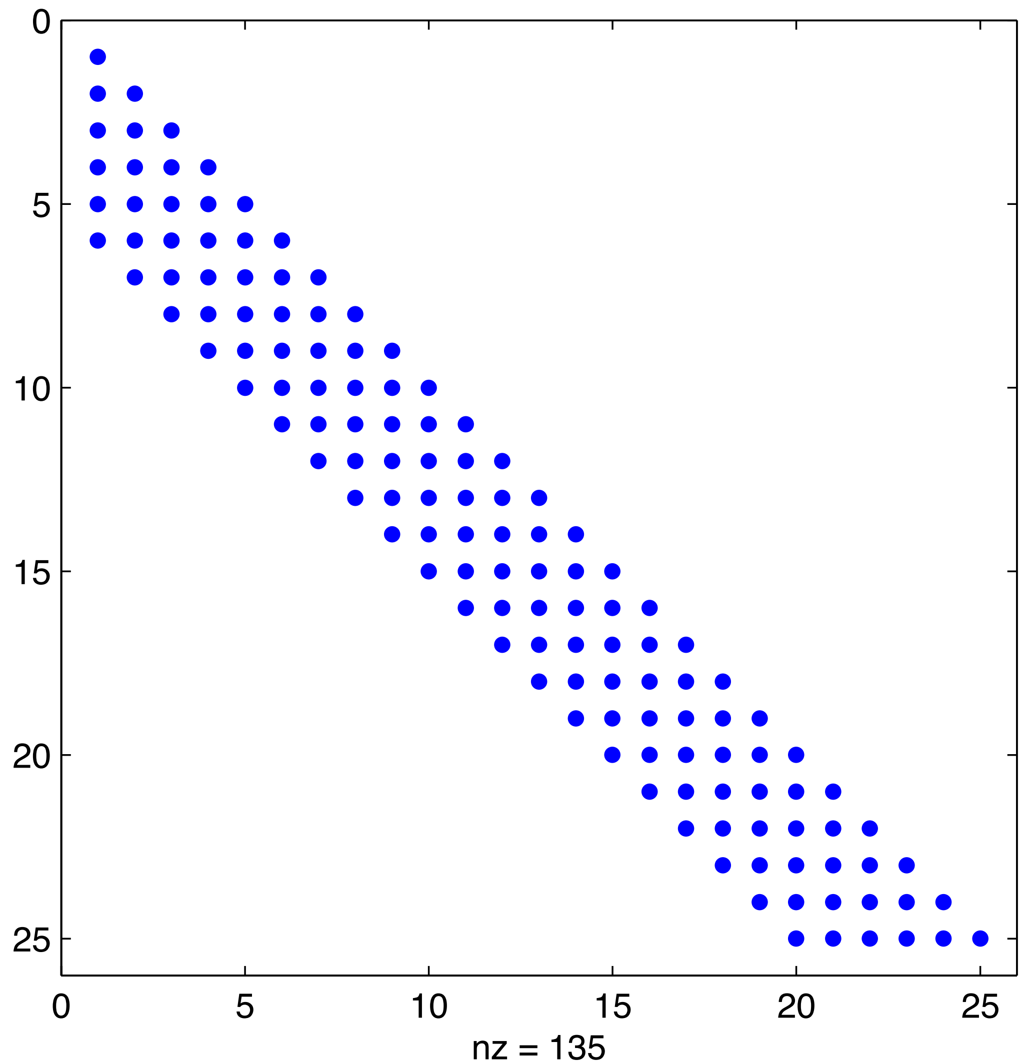
Bulge Chasing- The algorithm



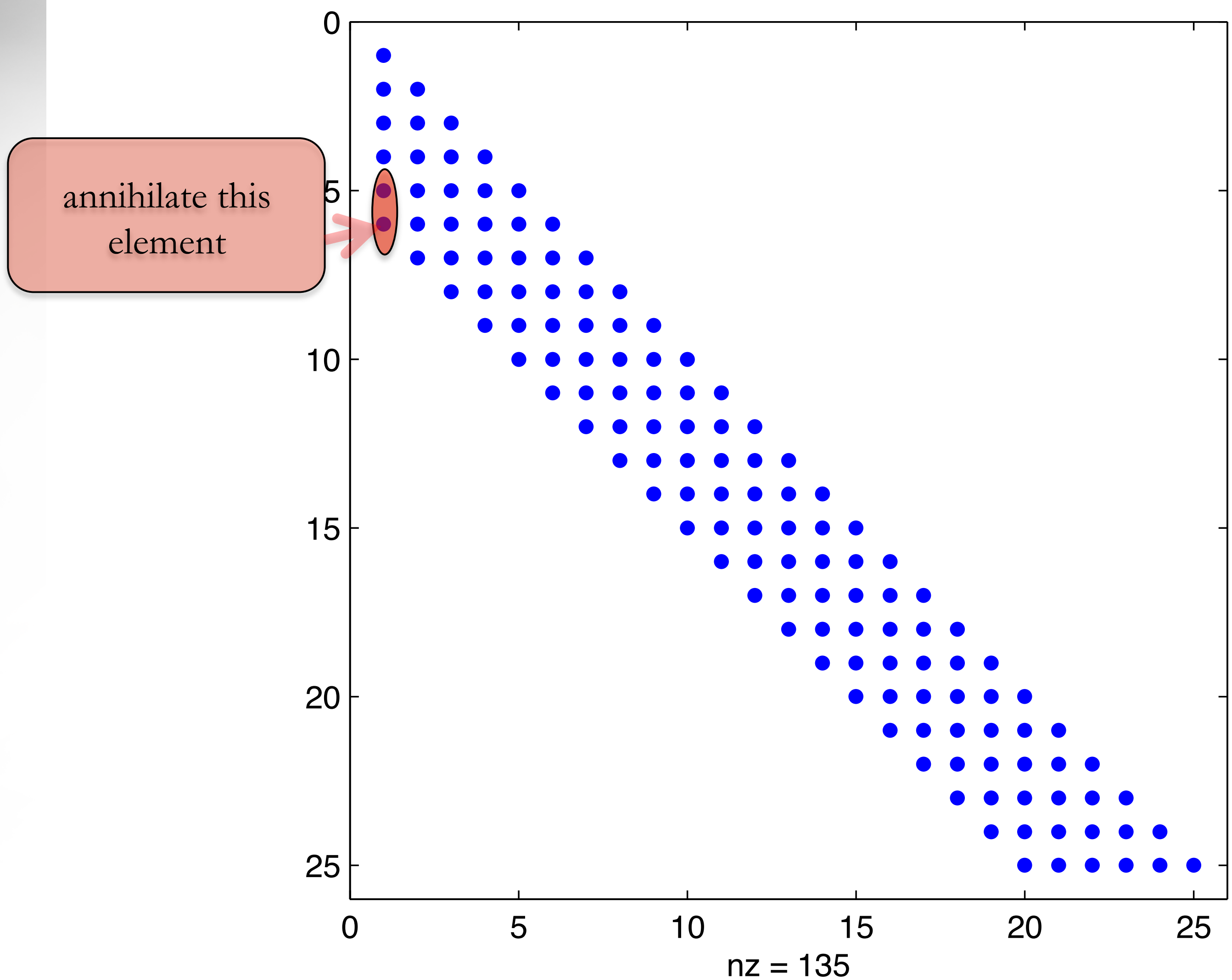
Bulge Chasing- The algorithm

don't be scared,
We can redesign the algo in a new fashion.
nice shape and color ☺

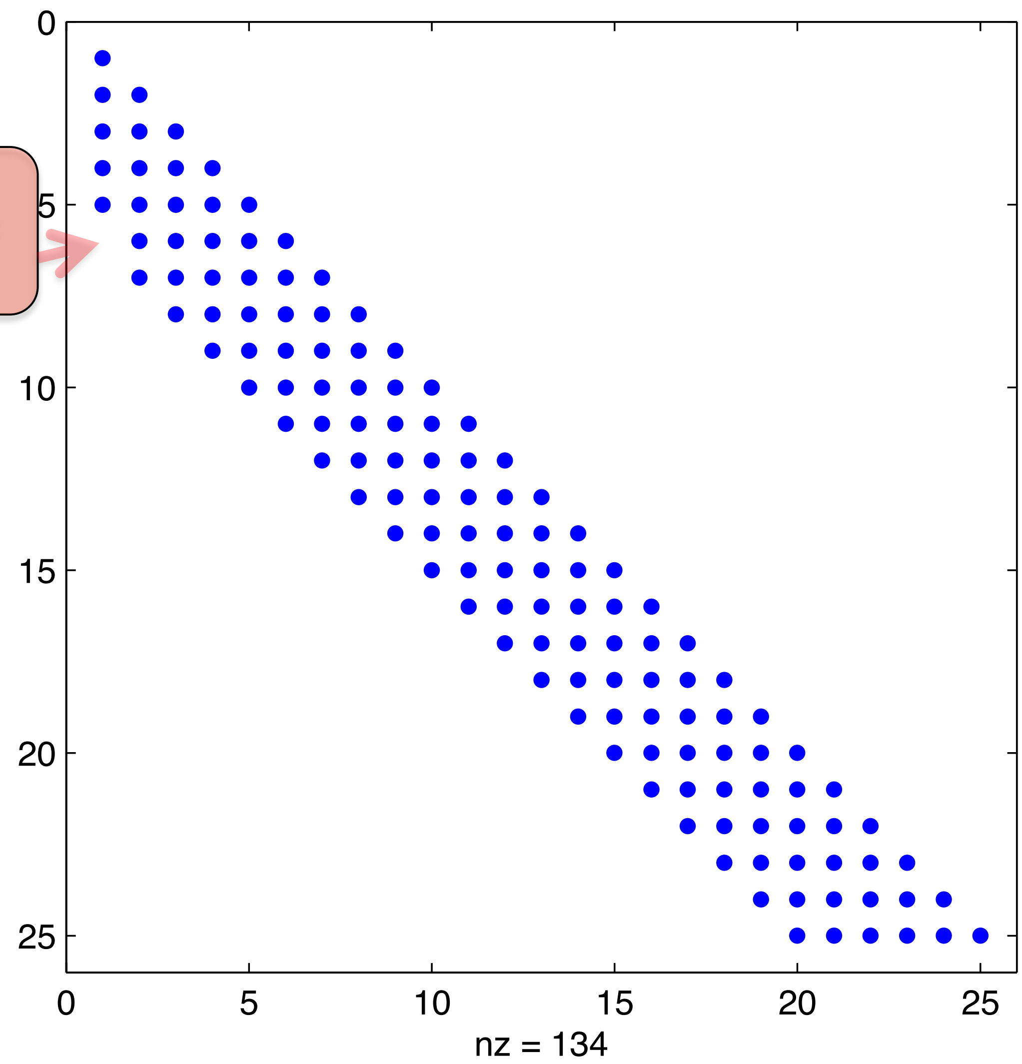
The Bulge chasing algorithm, step -2-



The Bulge chasing algorithm, step -2-



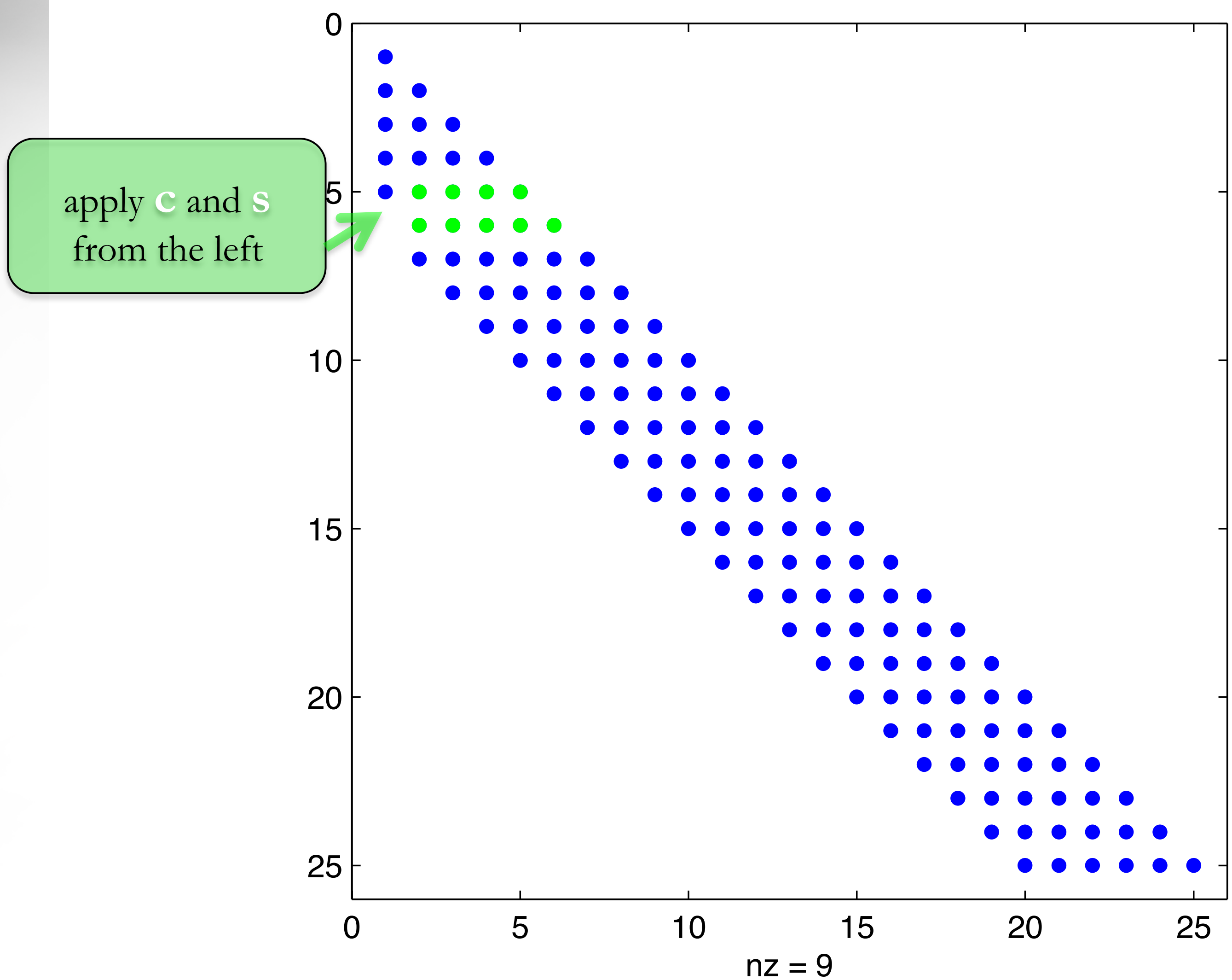
The Bulge chasing algorithm, step -2-



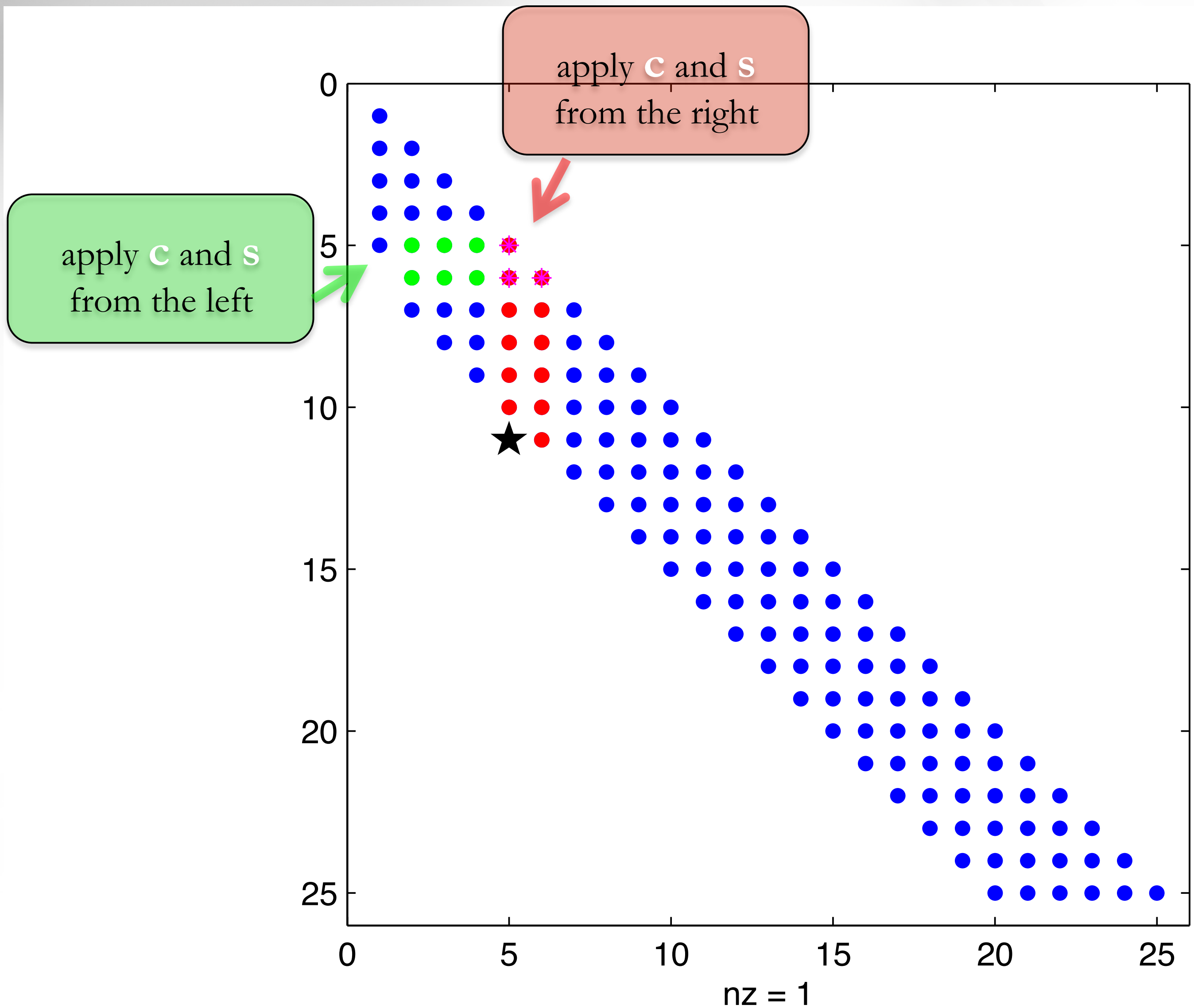
We generate a Givens rotation:

c, s

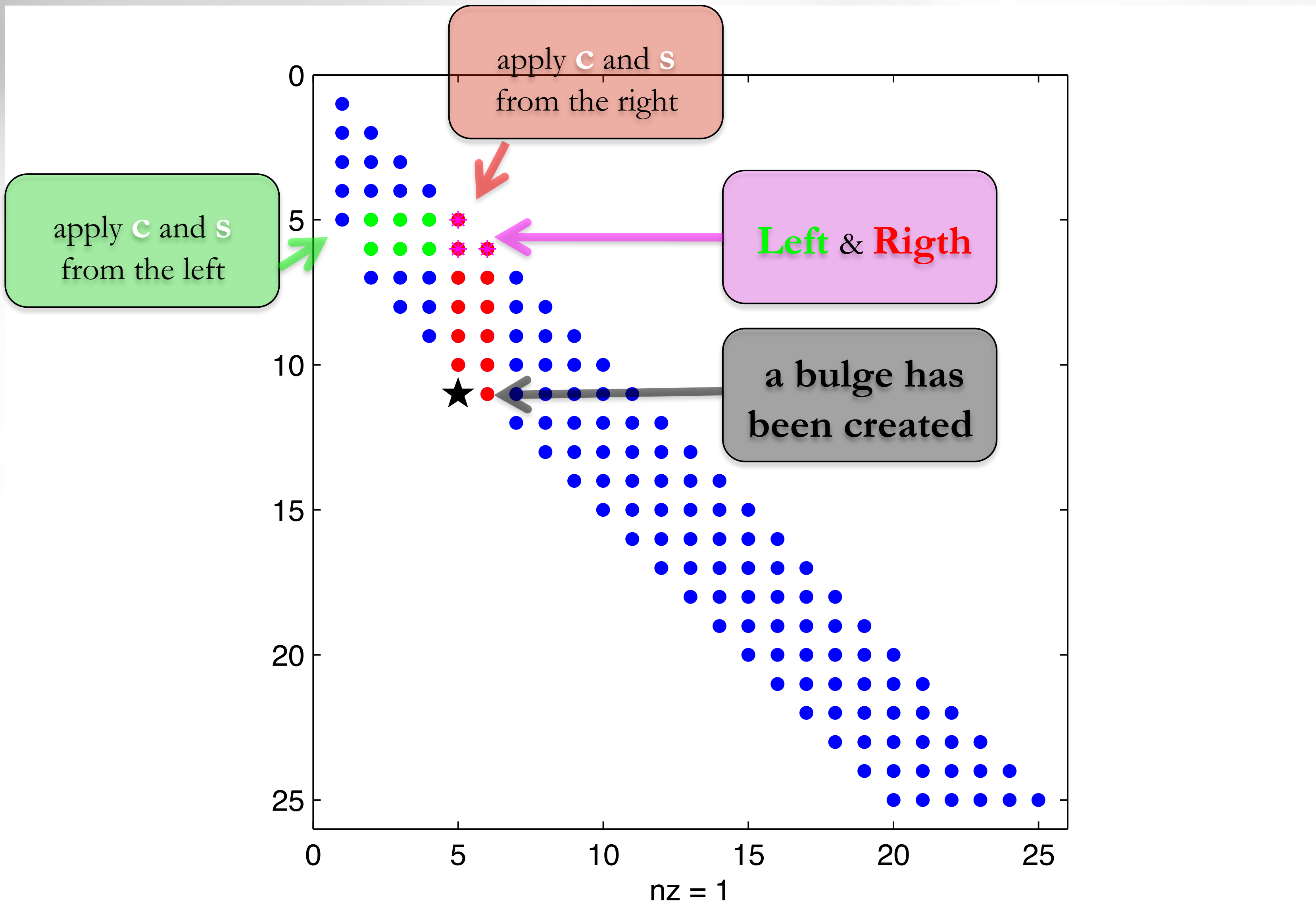
The Bulge chasing algorithm, step -2-



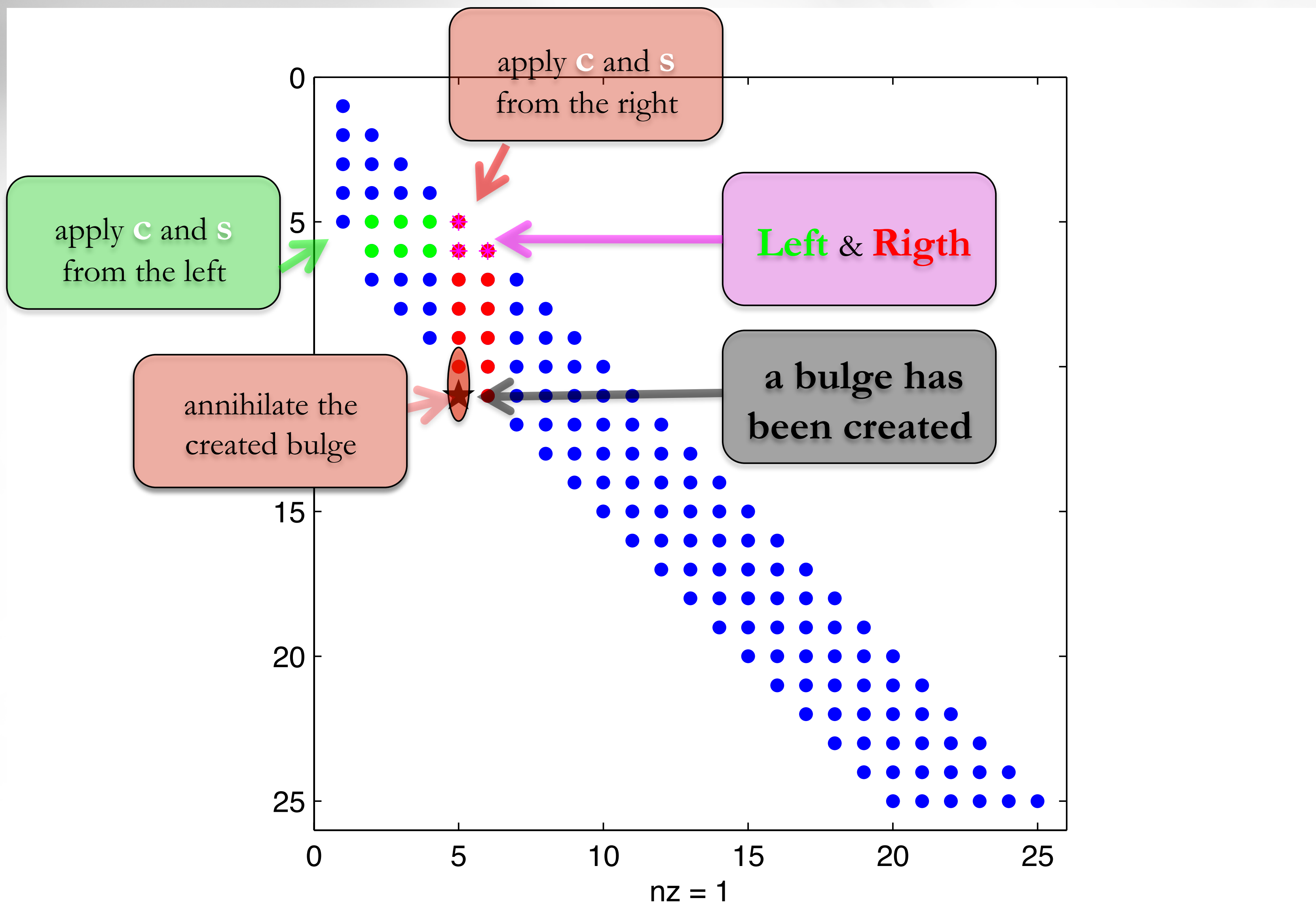
The Bulge chasing algorithm, step -2-



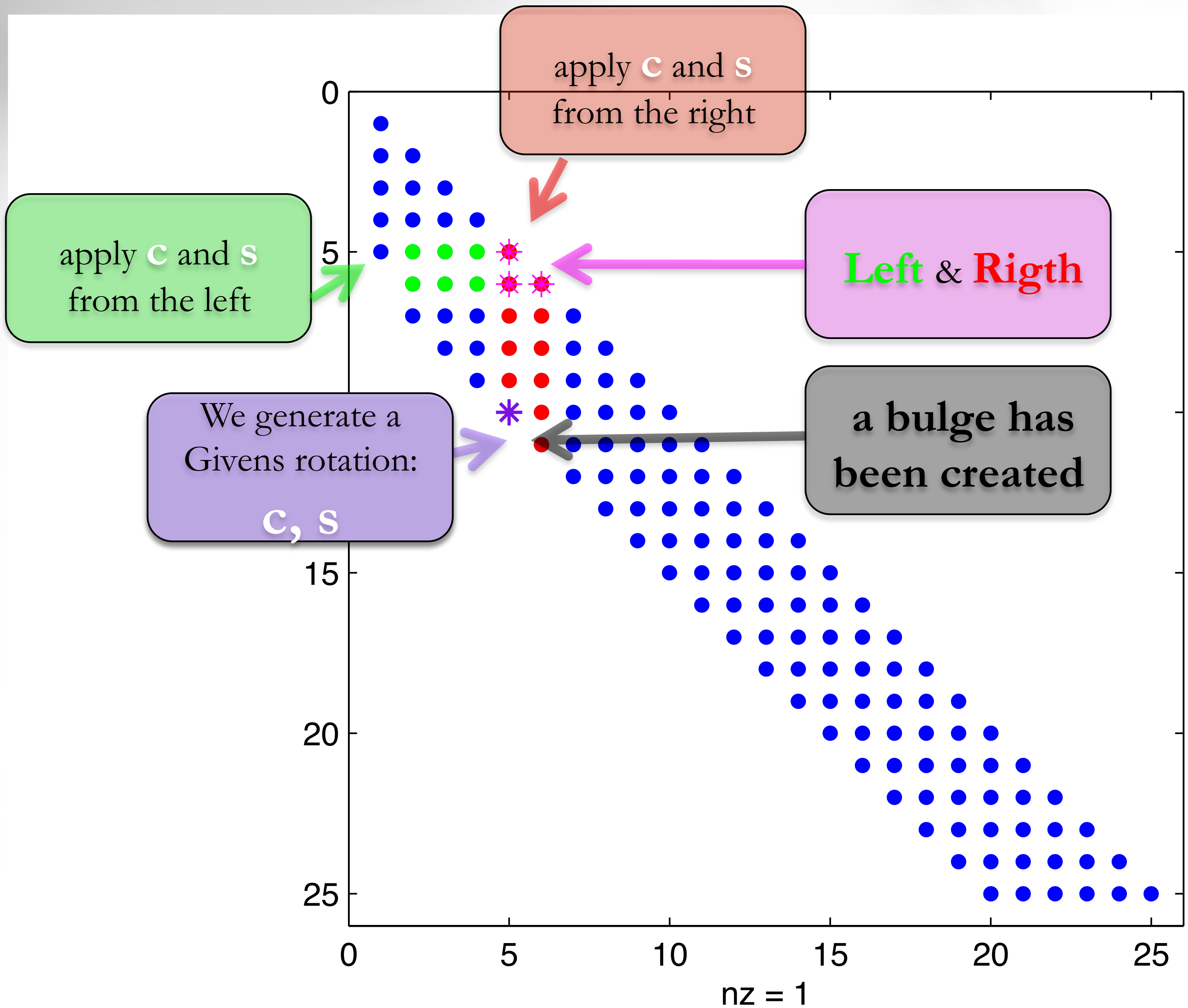
The Bulge chasing algorithm, step -2-



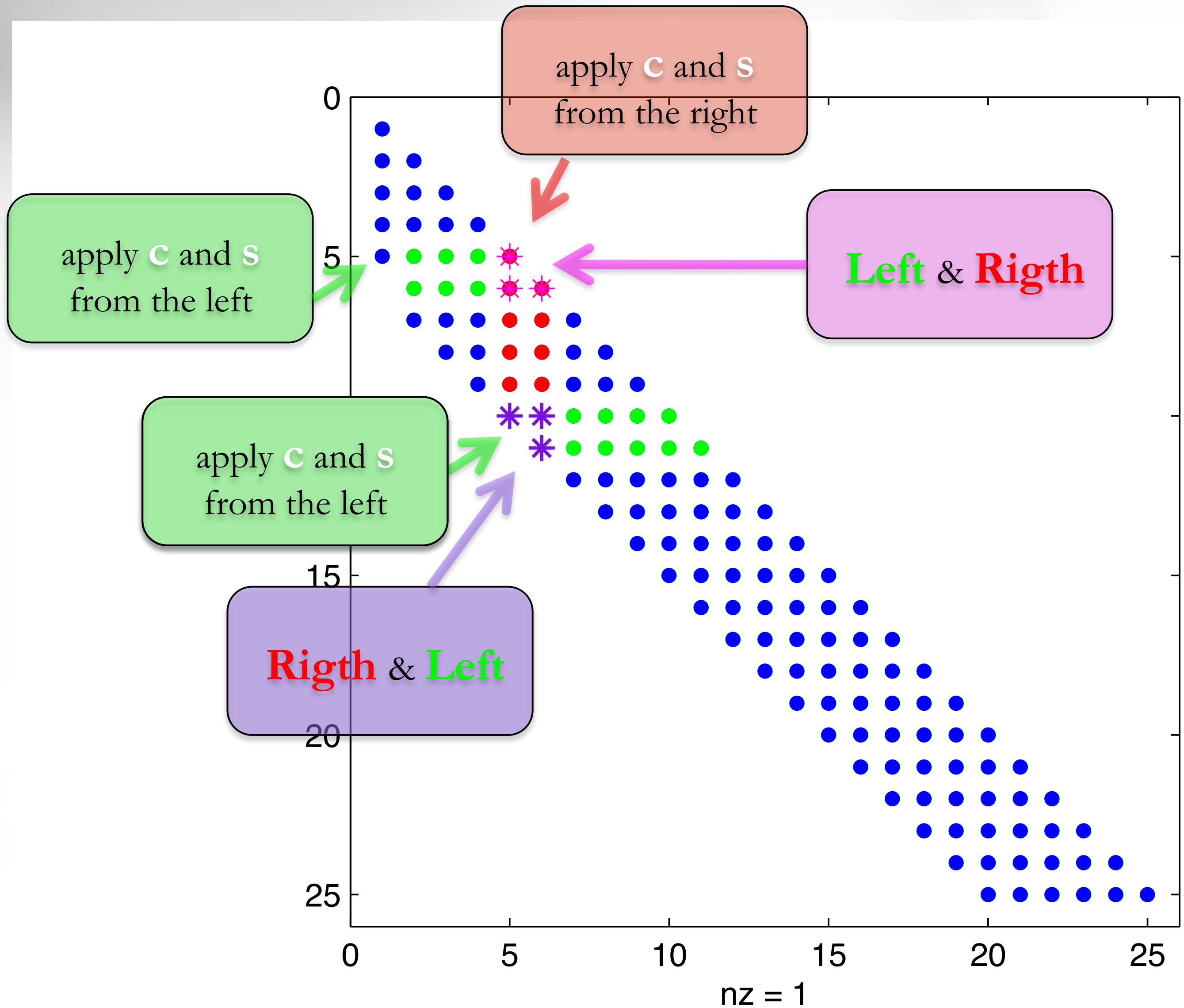
The Bulge chasing algorithm, step -2-



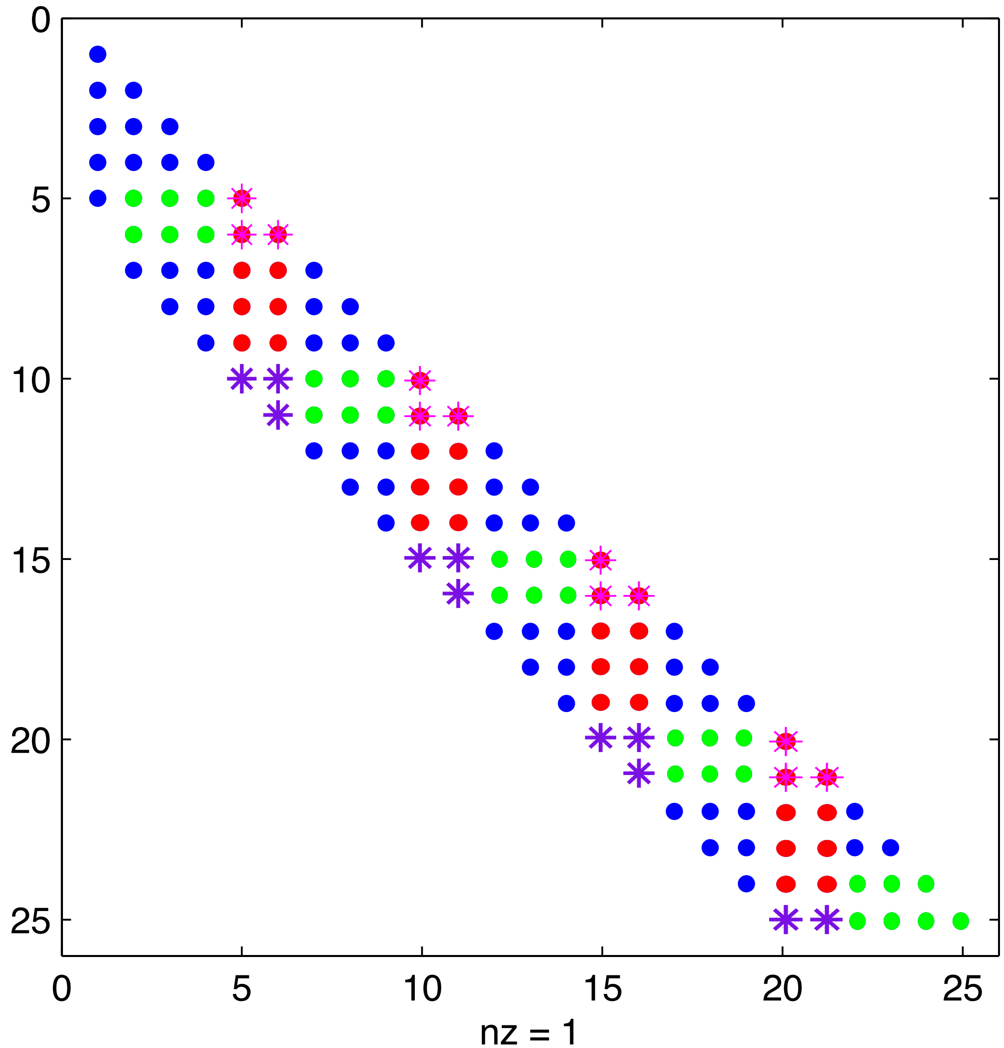
The Bulge chasing algorithm, step -2-



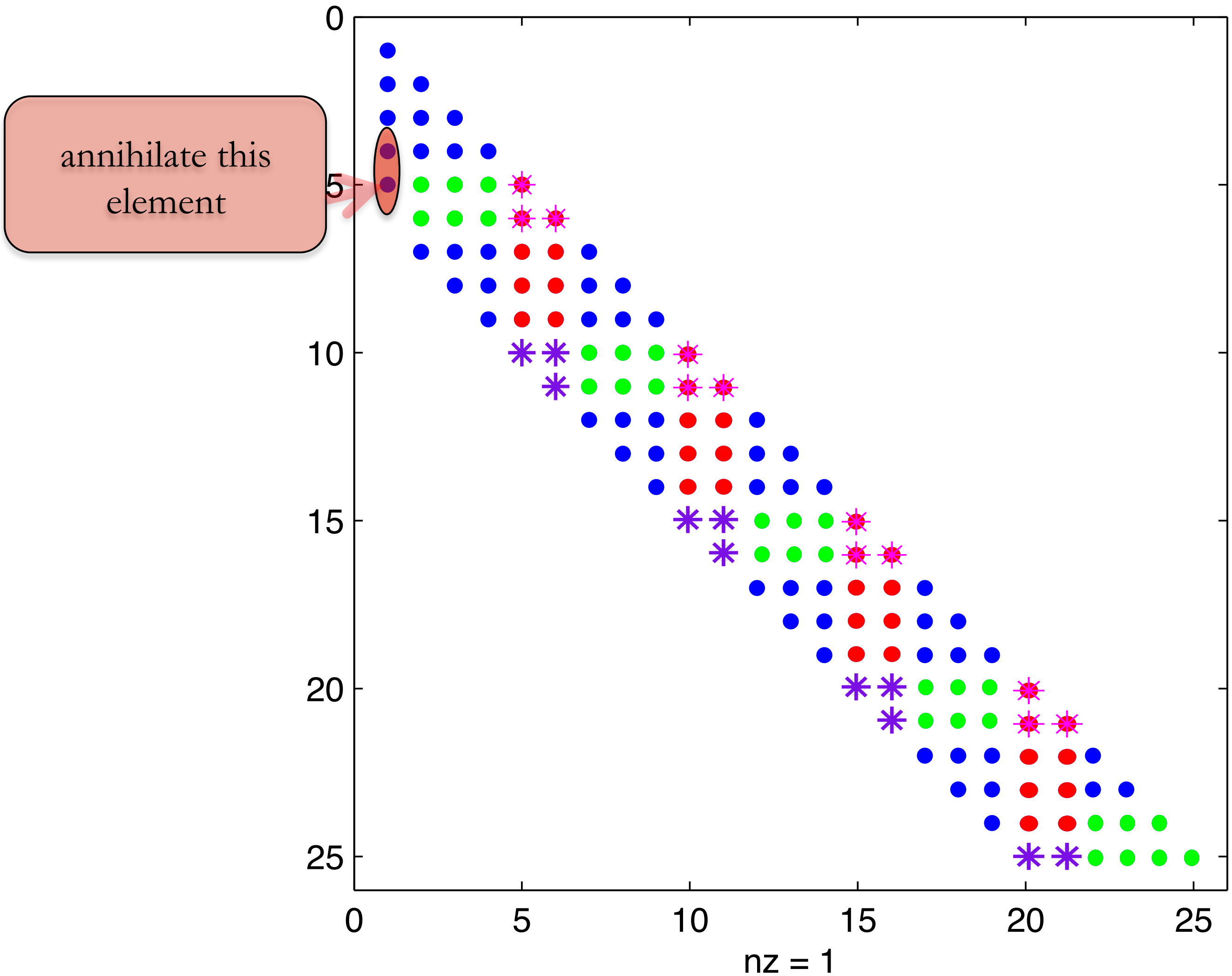
The Bulge chasing algorithm, step -2-



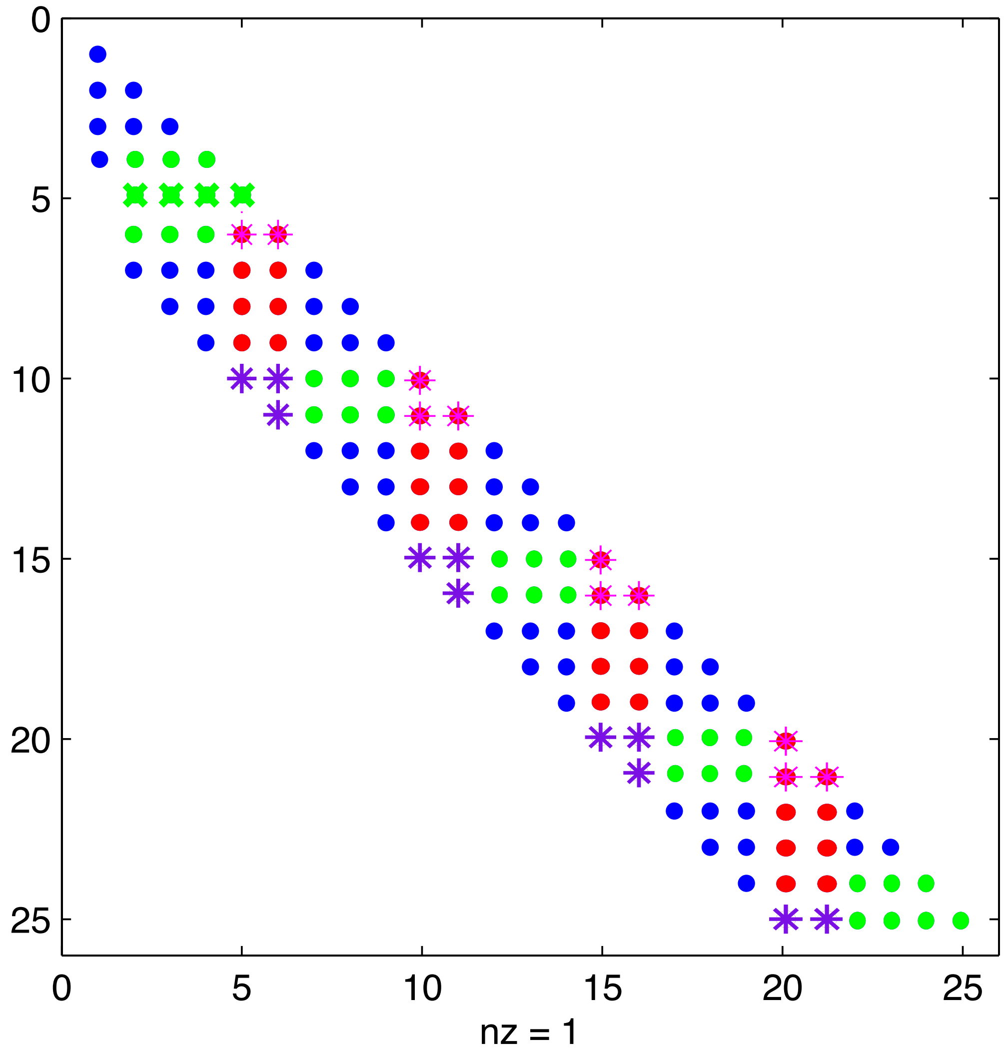
The Bulge chasing algorithm, step -2-



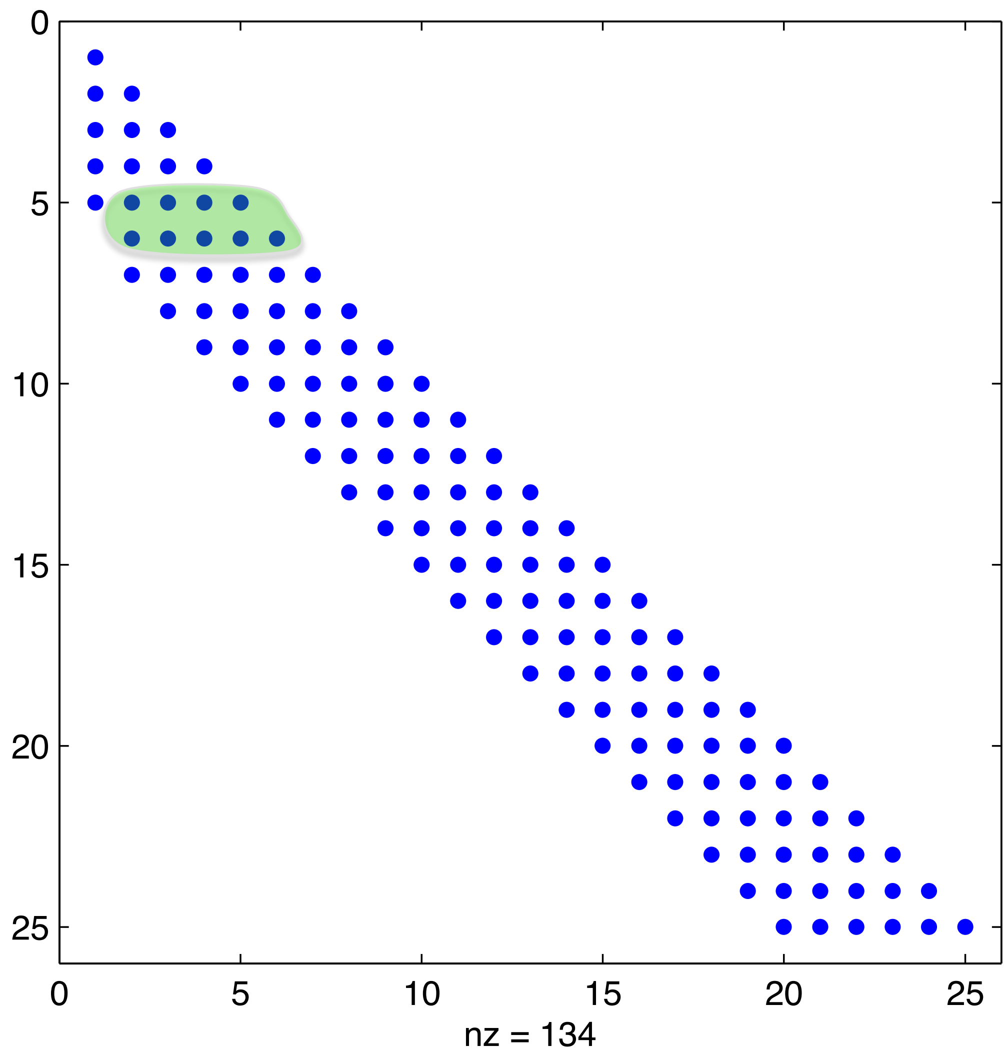
The Bulge chasing algorithm, step -2-



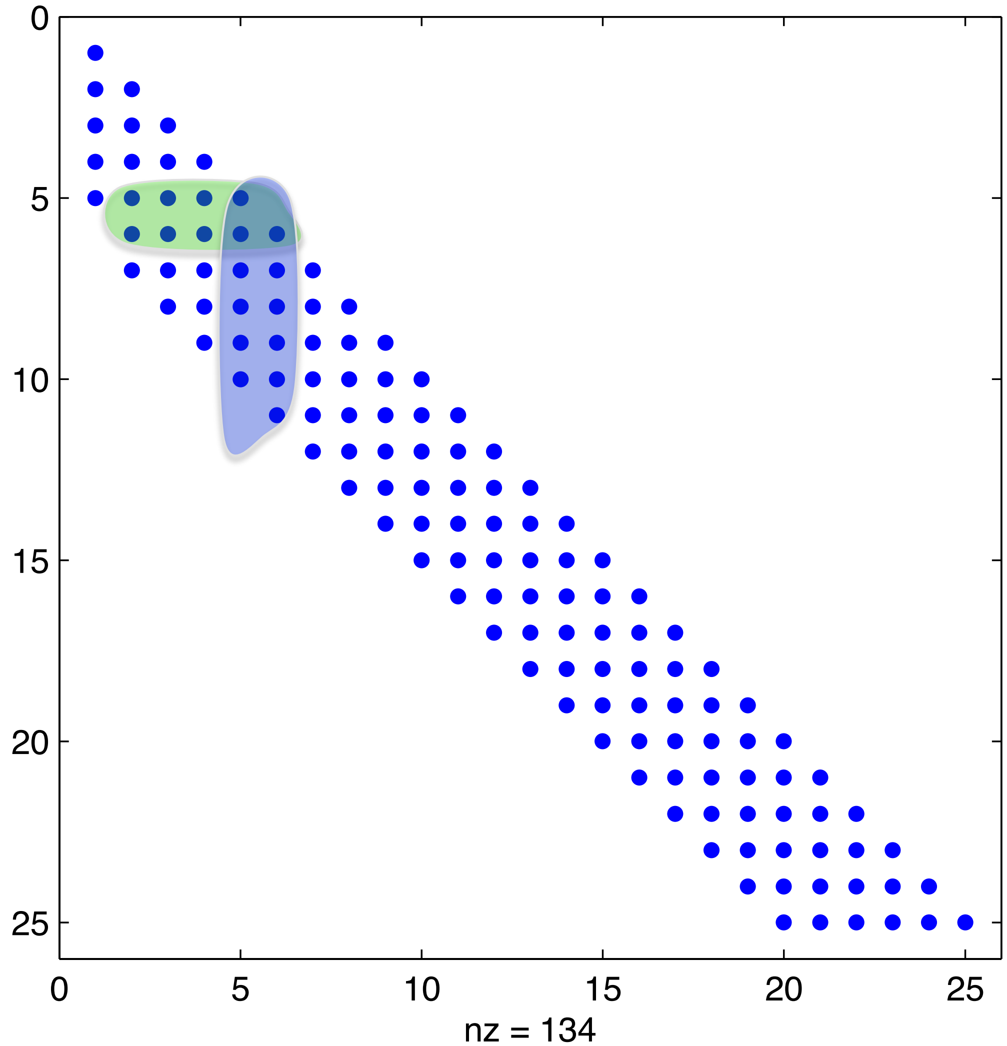
The Bulge chasing algorithm, step -2-



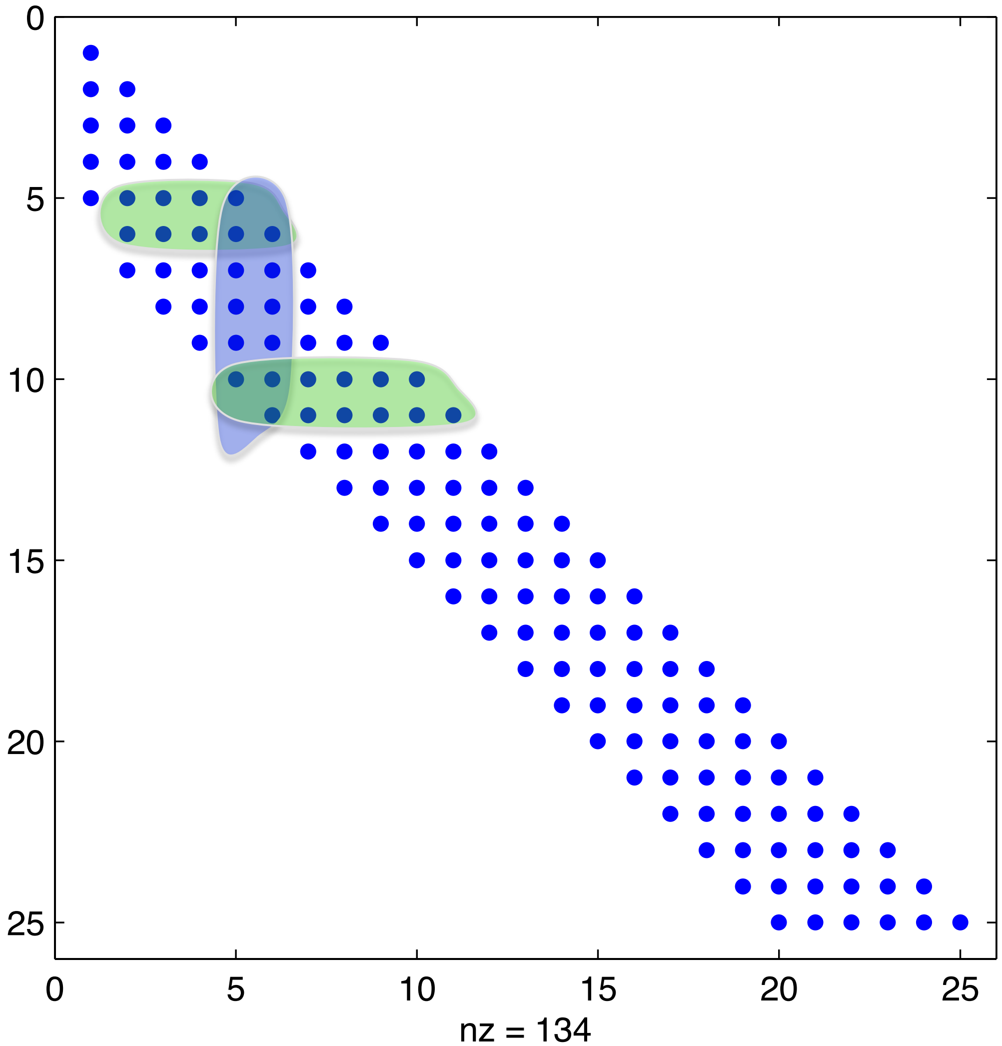
The Bulge chasing algorithm, step -2-



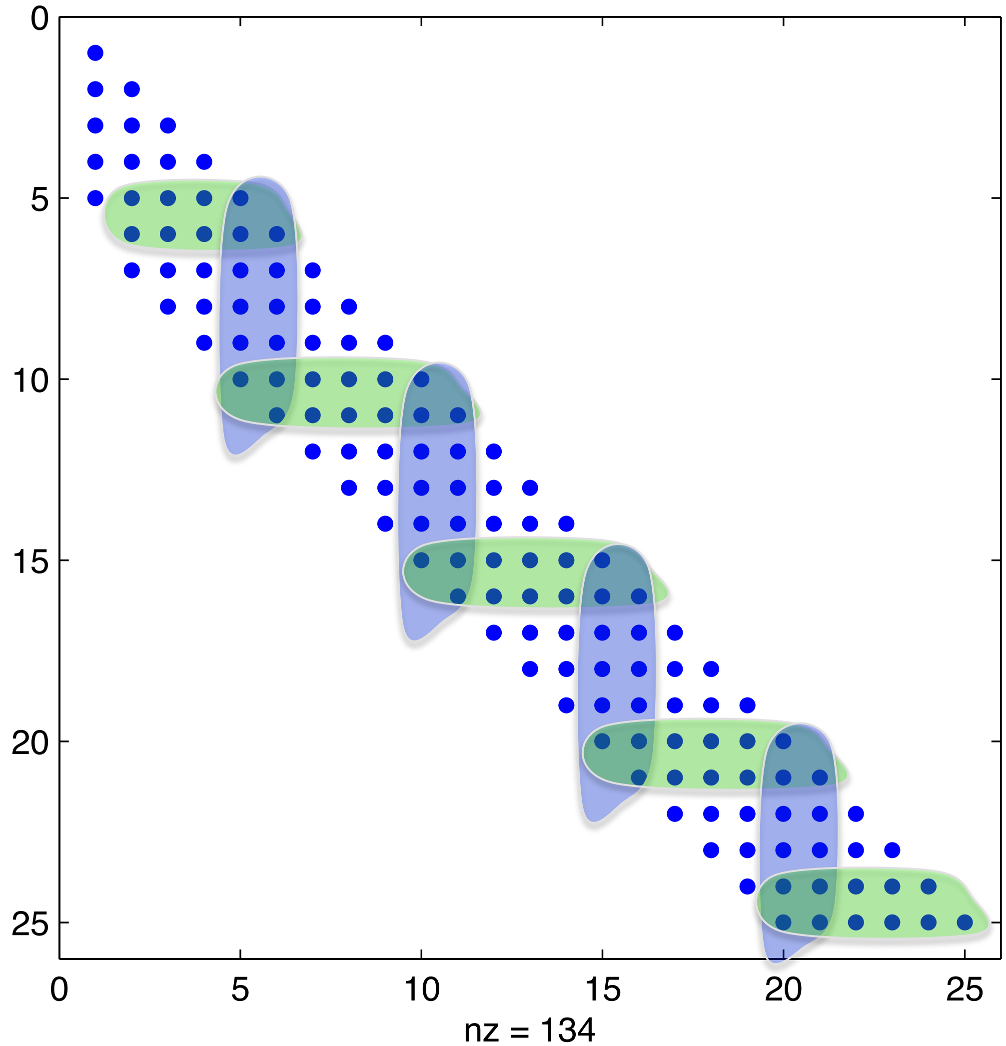
The Bulge chasing algorithm, step -2-



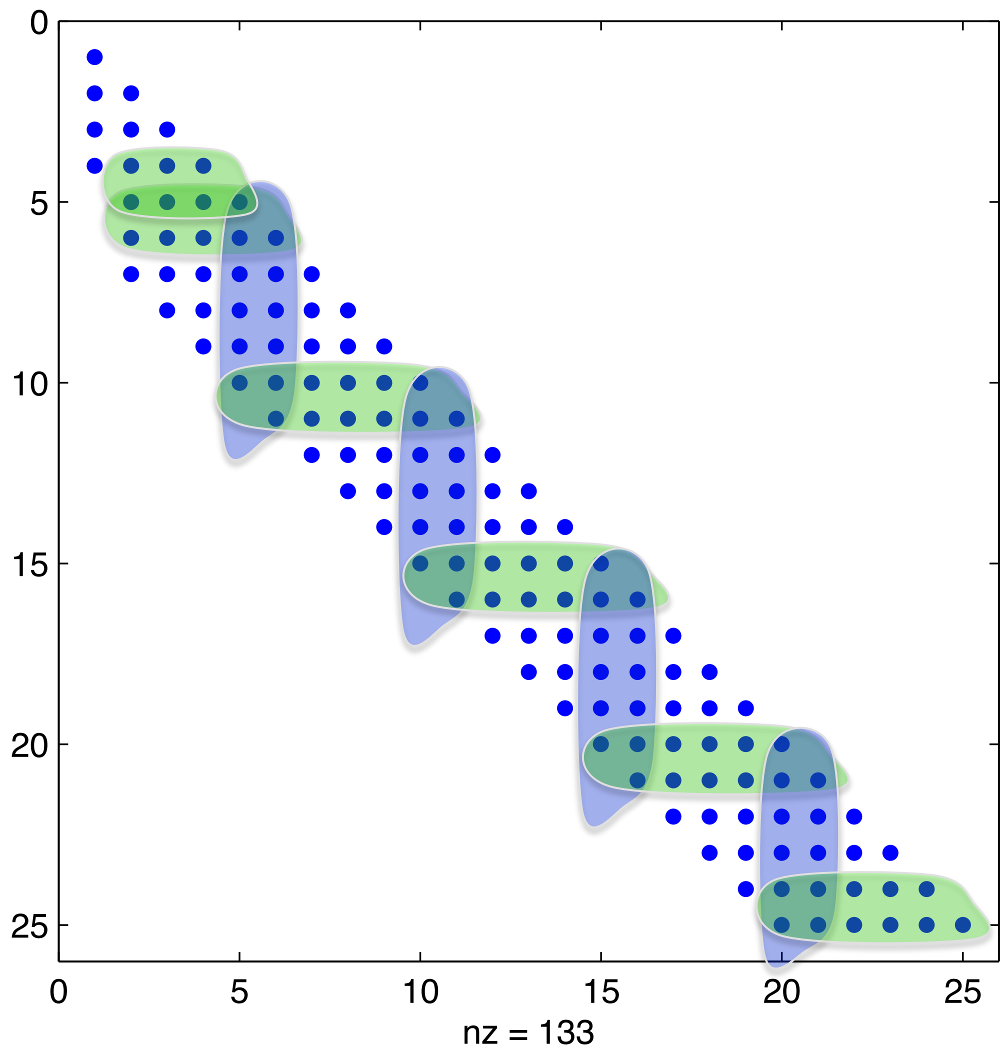
The Bulge chasing algorithm, step -2-



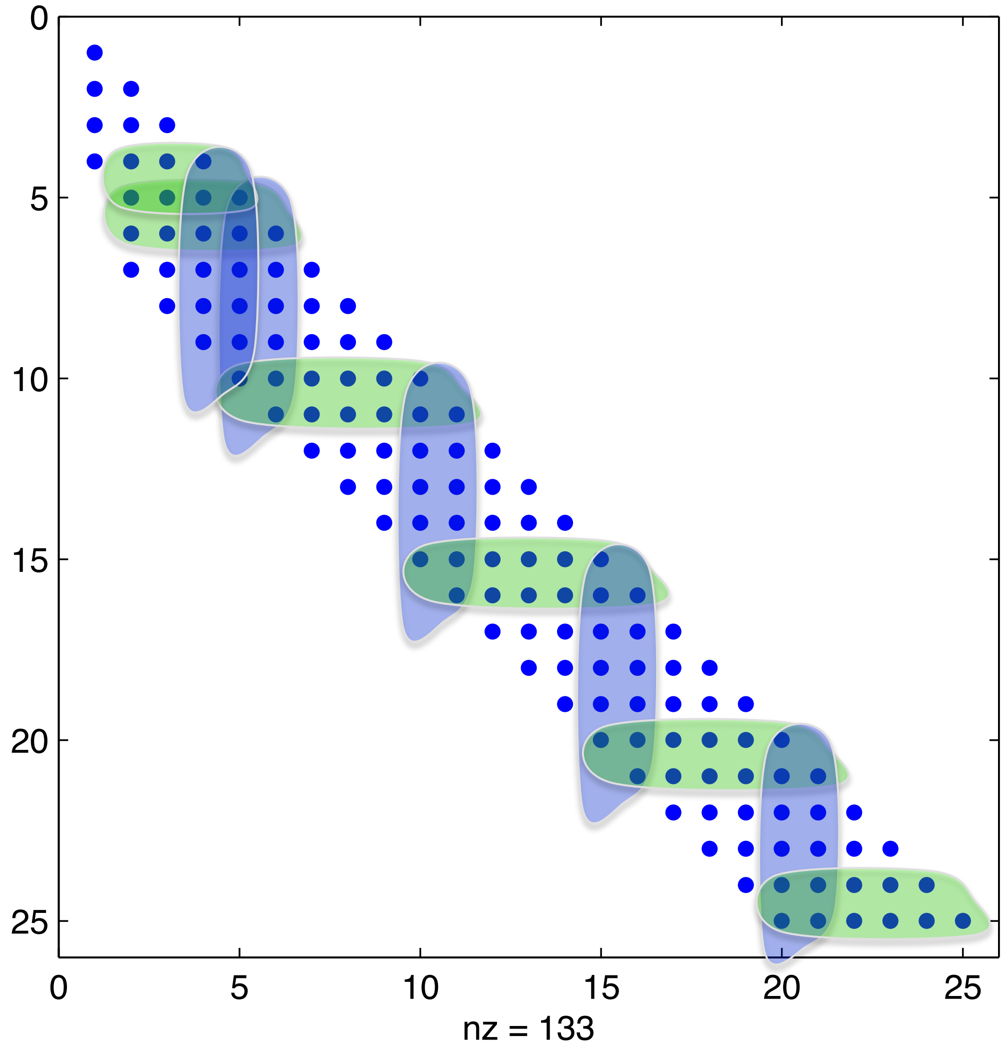
The Bulge chasing algorithm, step -2-



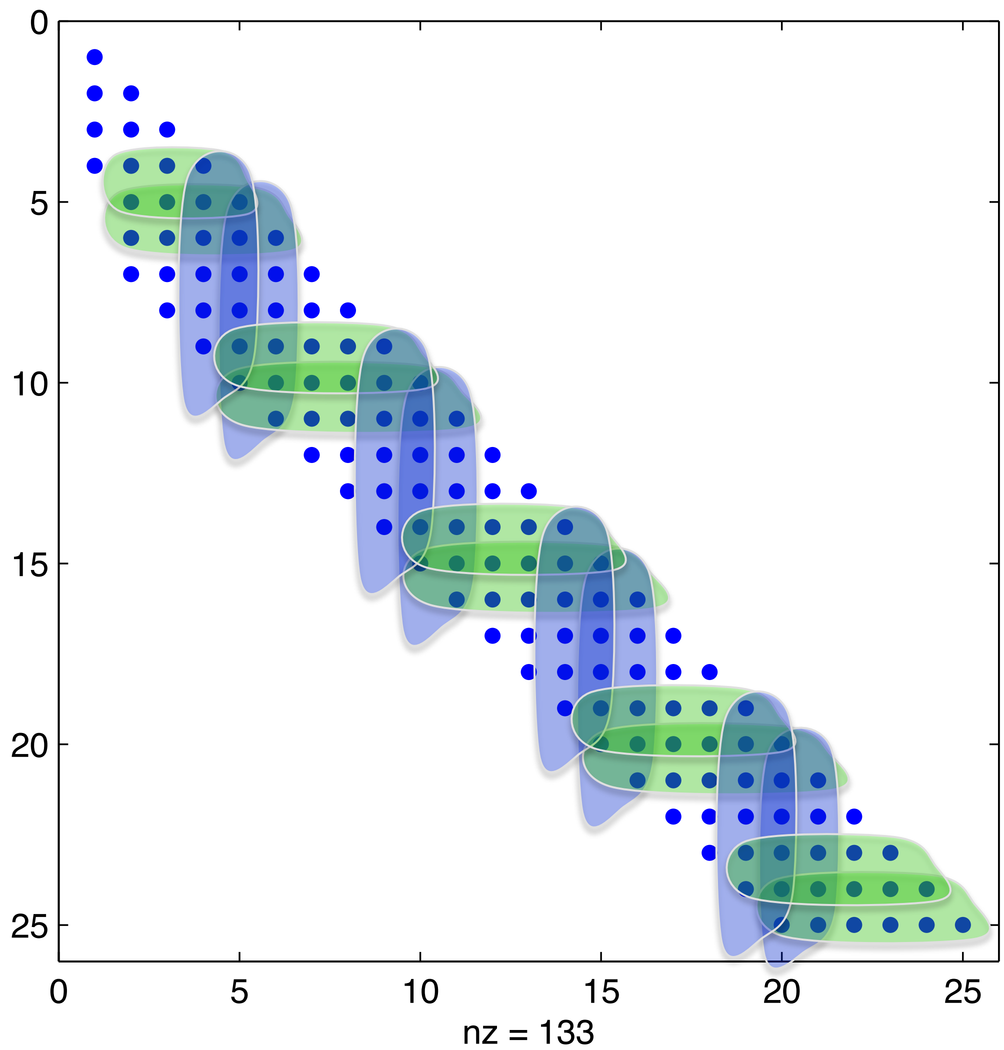
The Bulge chasing algorithm, step -2-



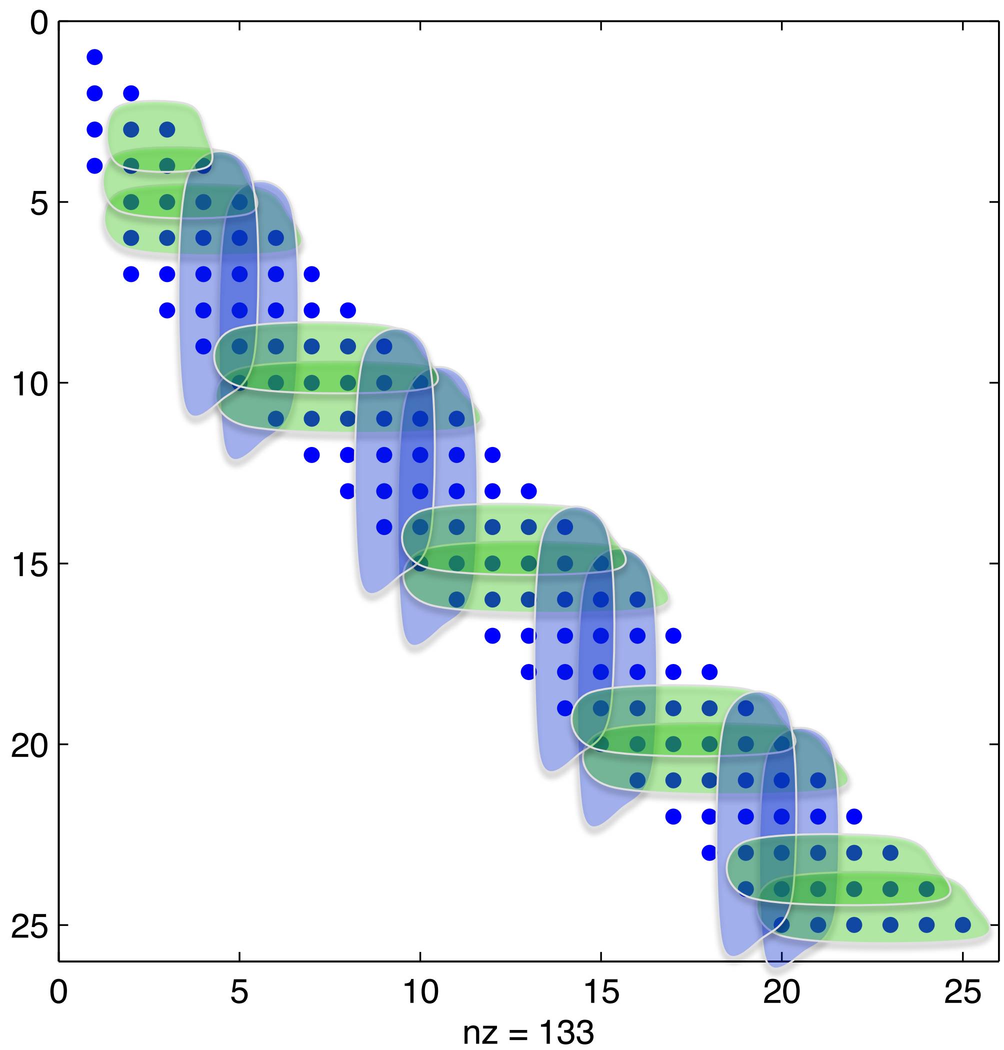
The Bulge chasing algorithm, step -2-



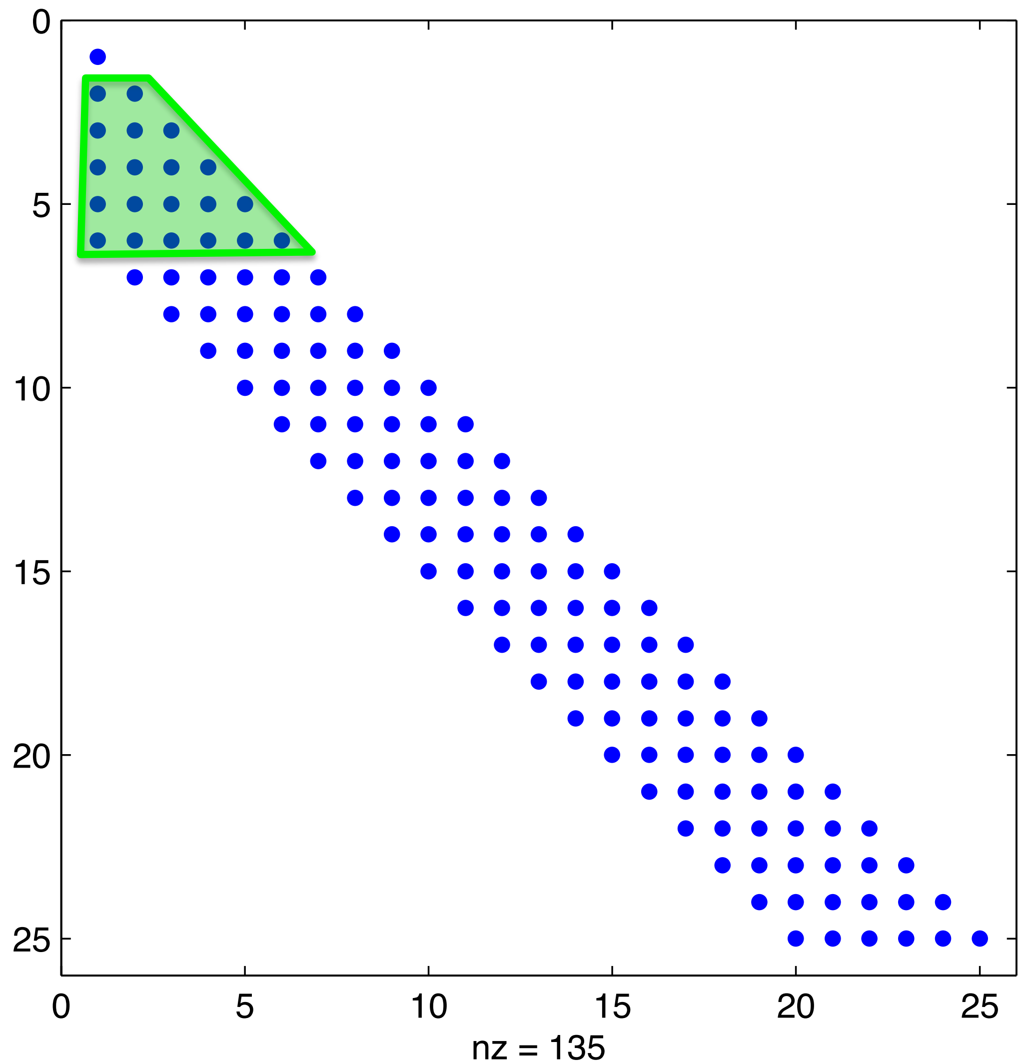
The Bulge chasing algorithm, step -2-



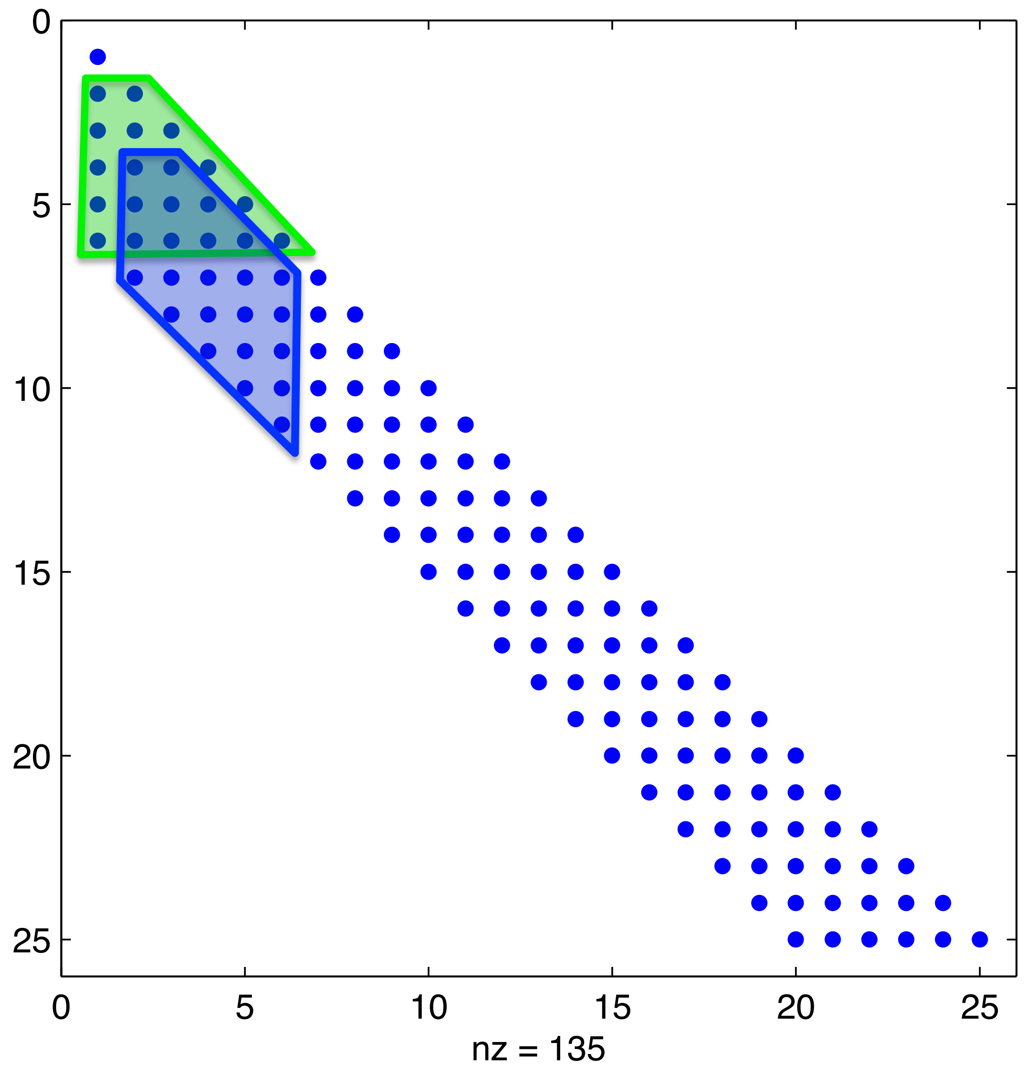
The Bulge chasing algorithm, step -2-



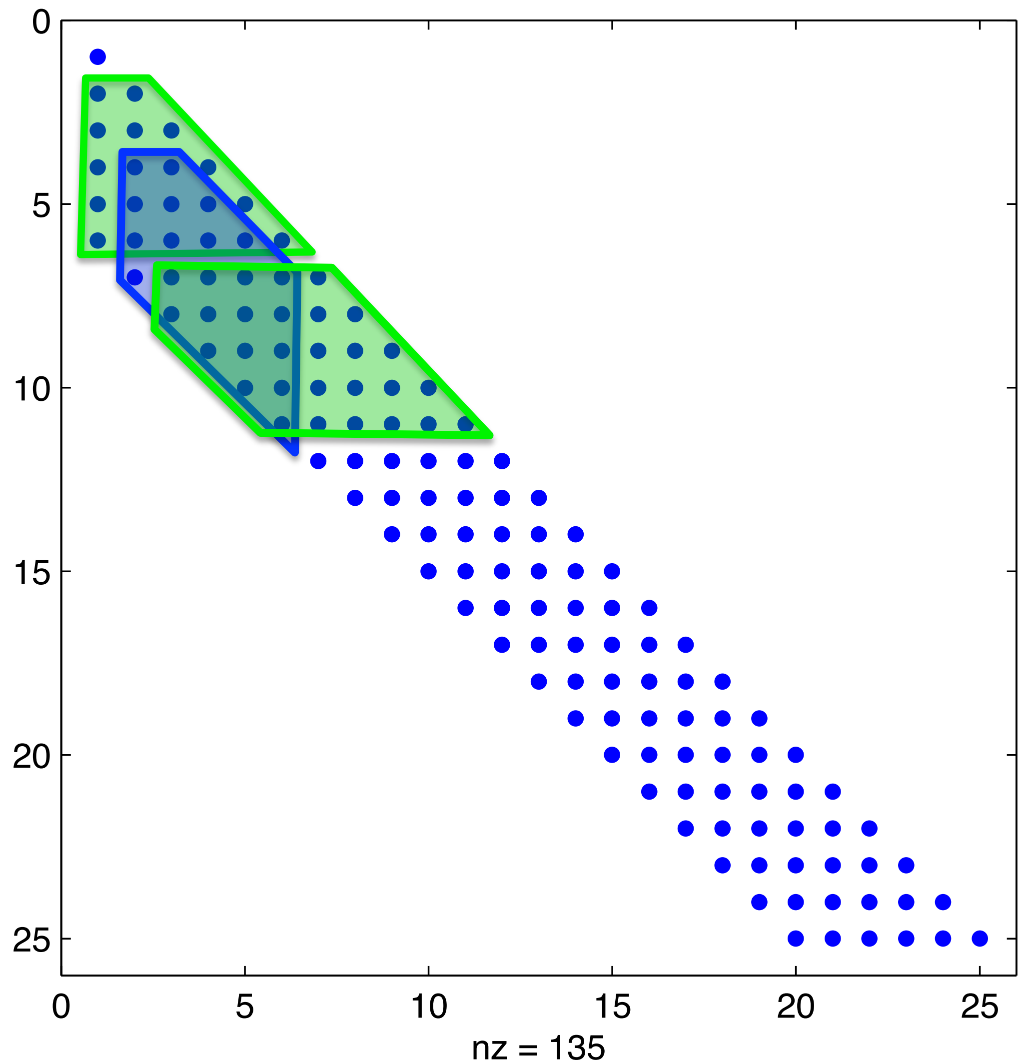
The Bulge chasing algorithm, step -2-



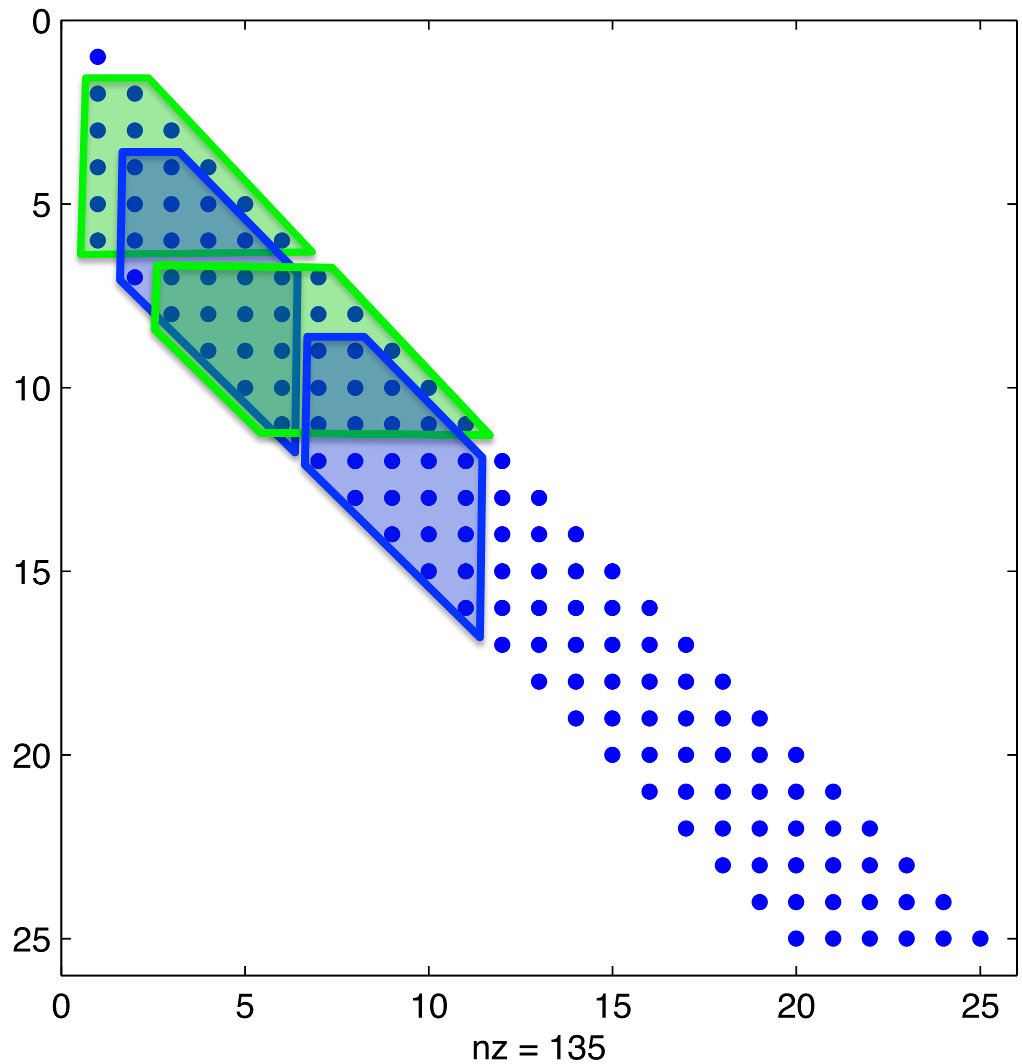
The Bulge chasing algorithm, step -2-



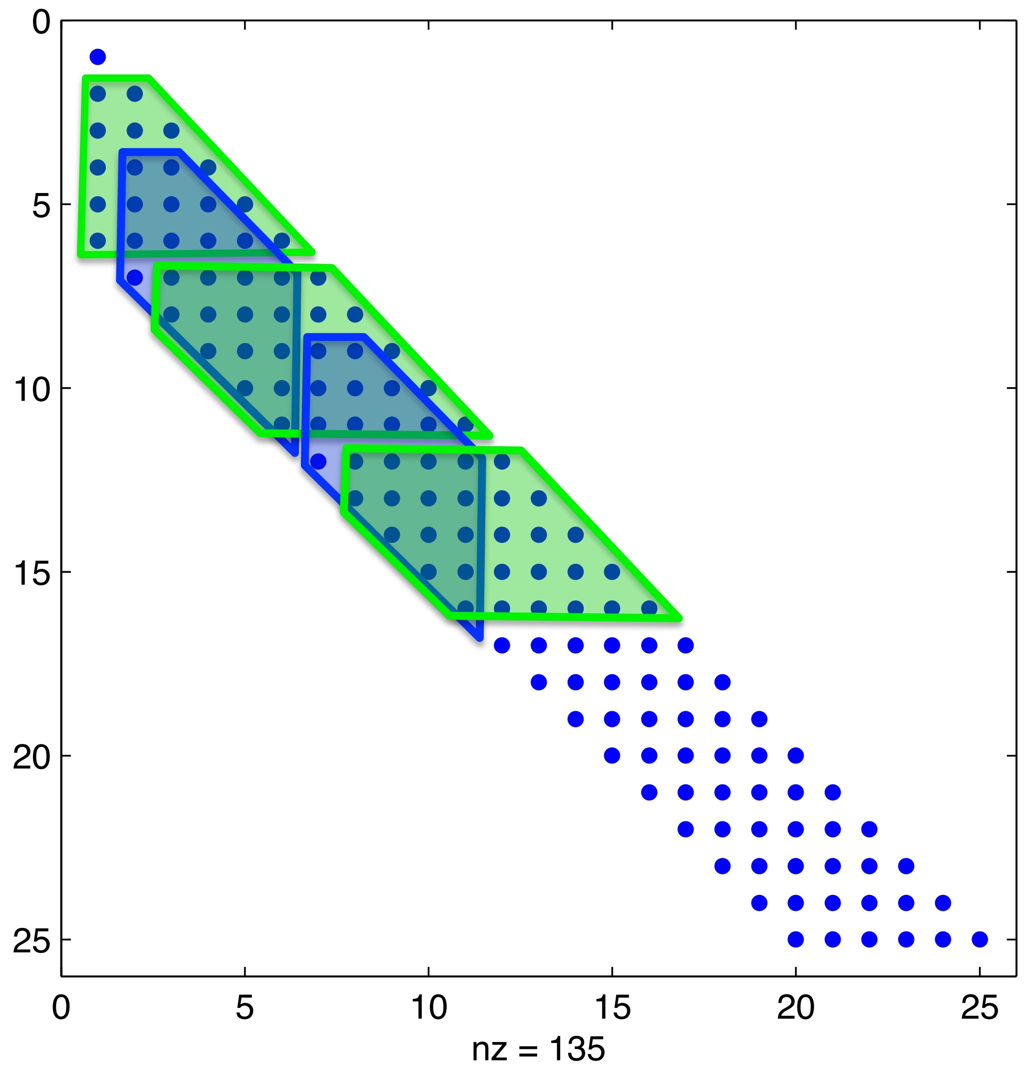
The Bulge chasing algorithm, step -2-



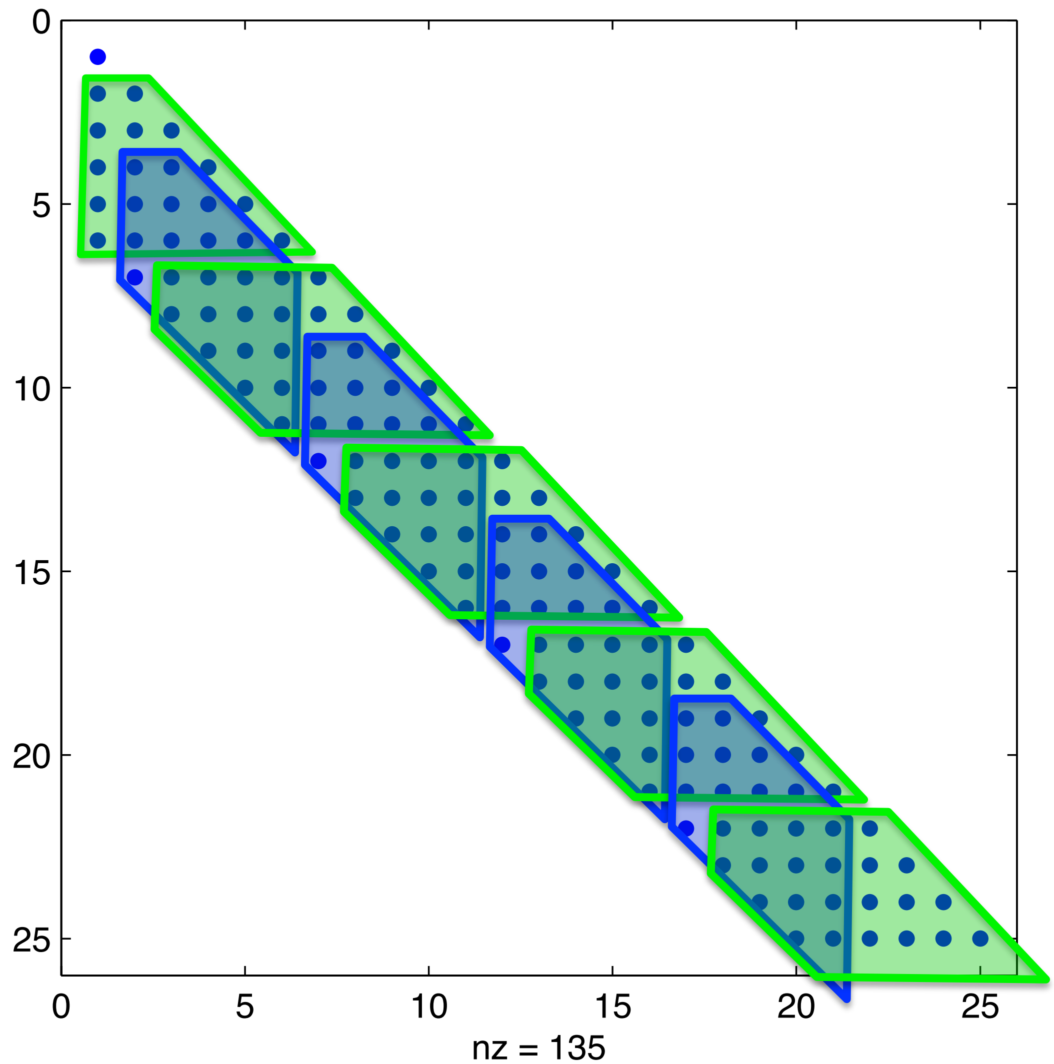
The Bulge chasing algorithm, step -2-



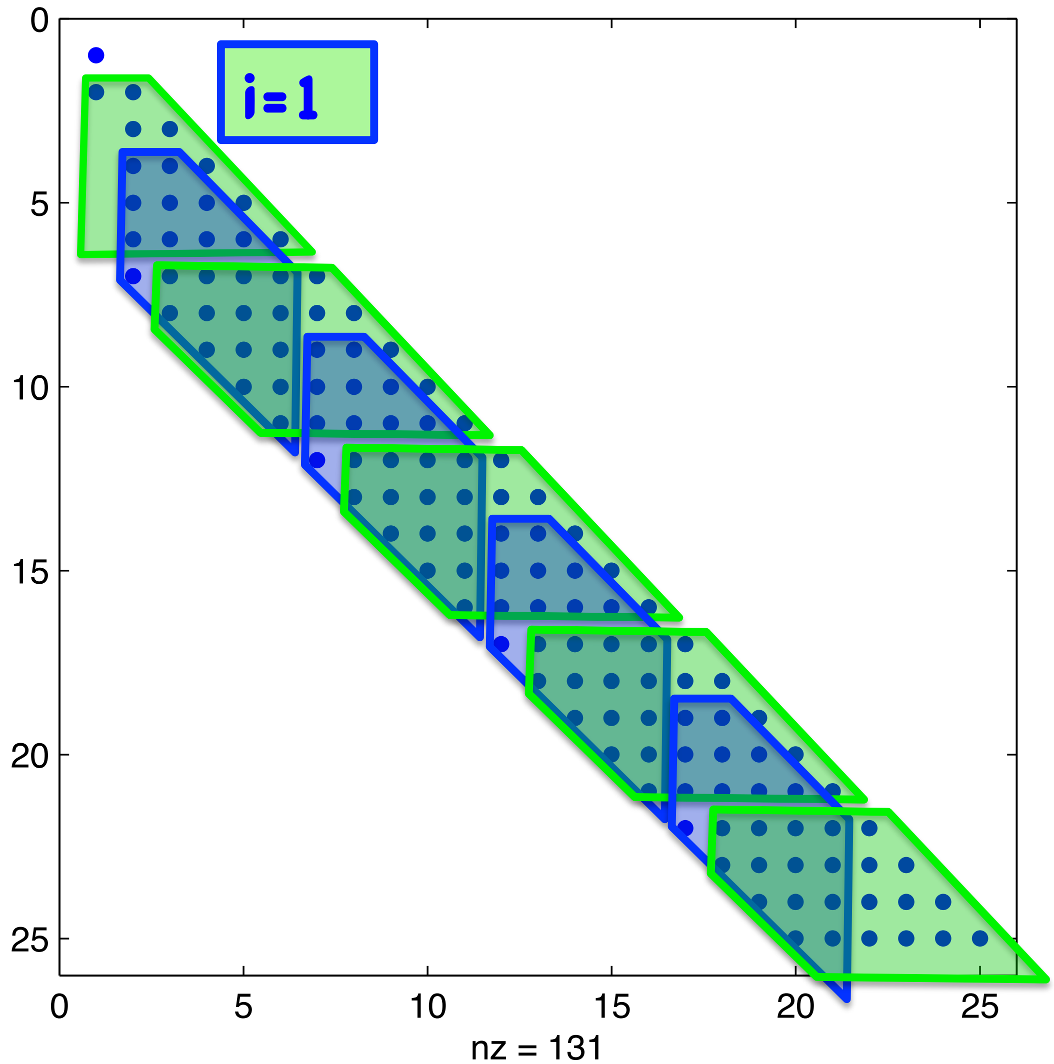
The Bulge chasing algorithm, step -2-



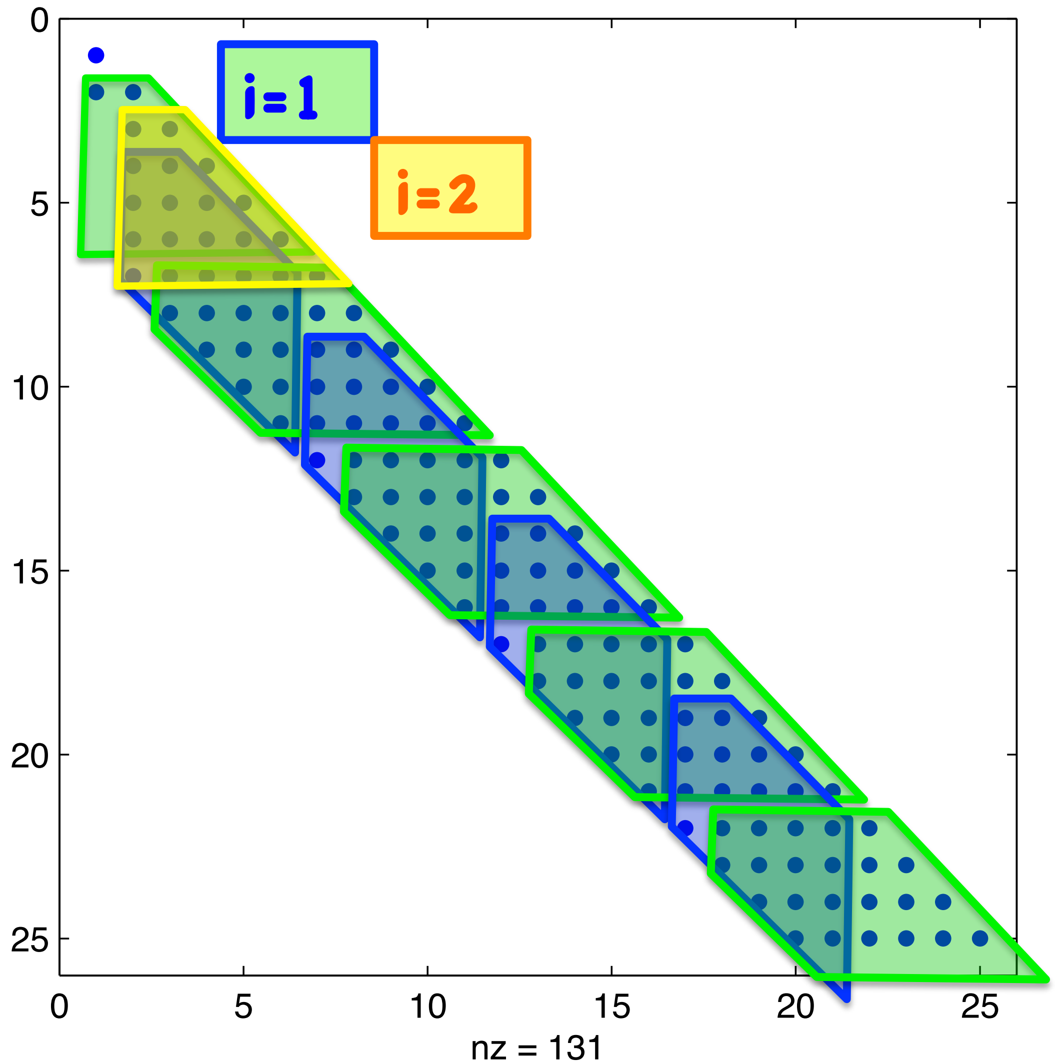
The Bulge chasing algorithm, step -2-



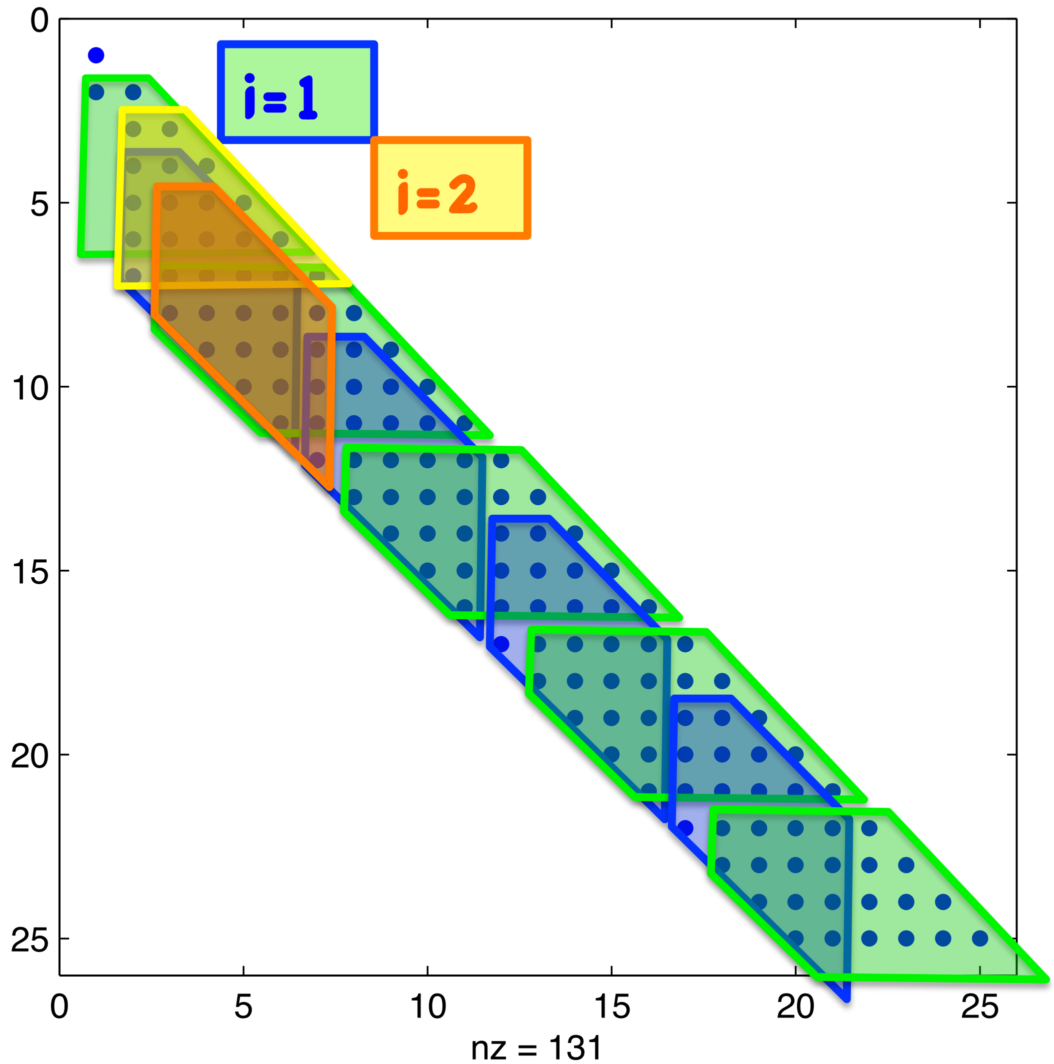
The Bulge chasing algorithm, step -2-



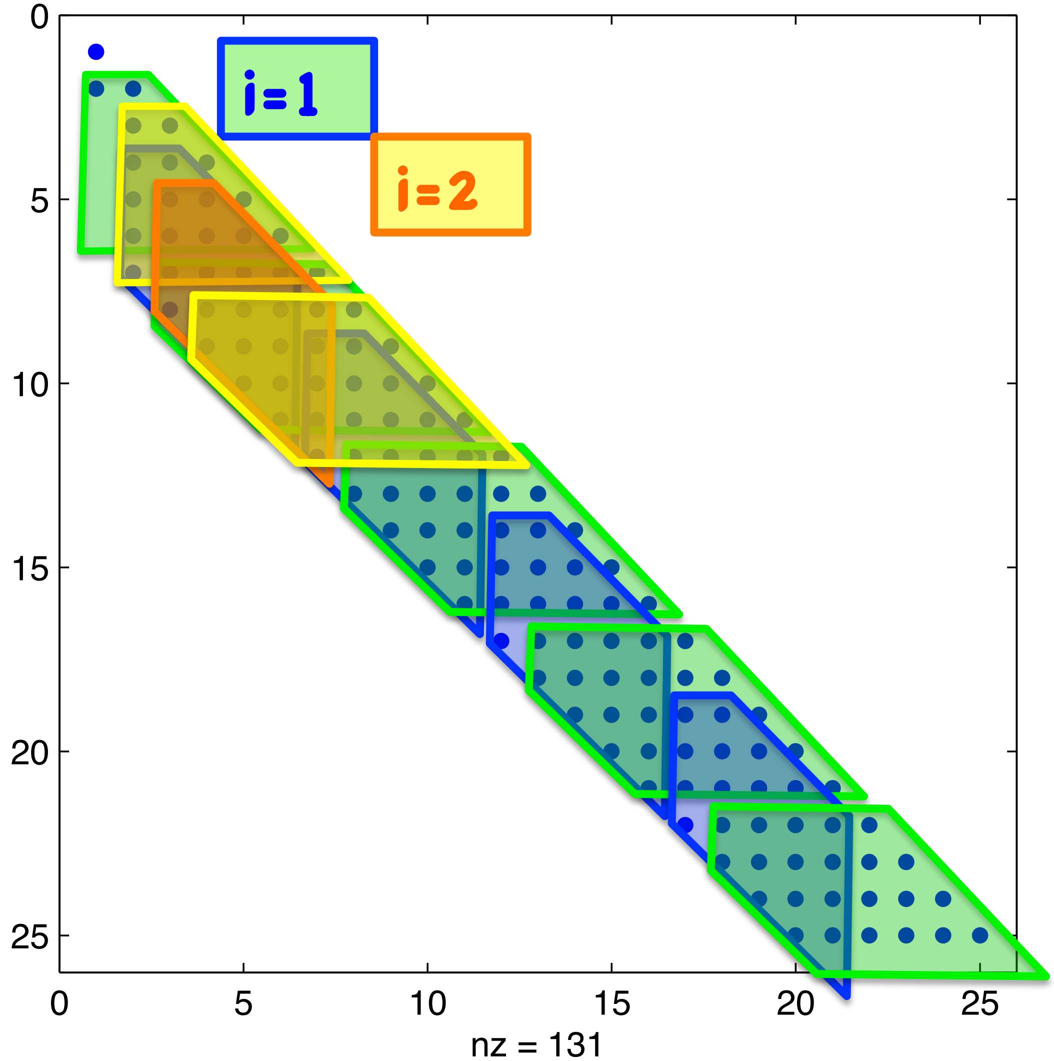
The Bulge chasing algorithm, step -2-



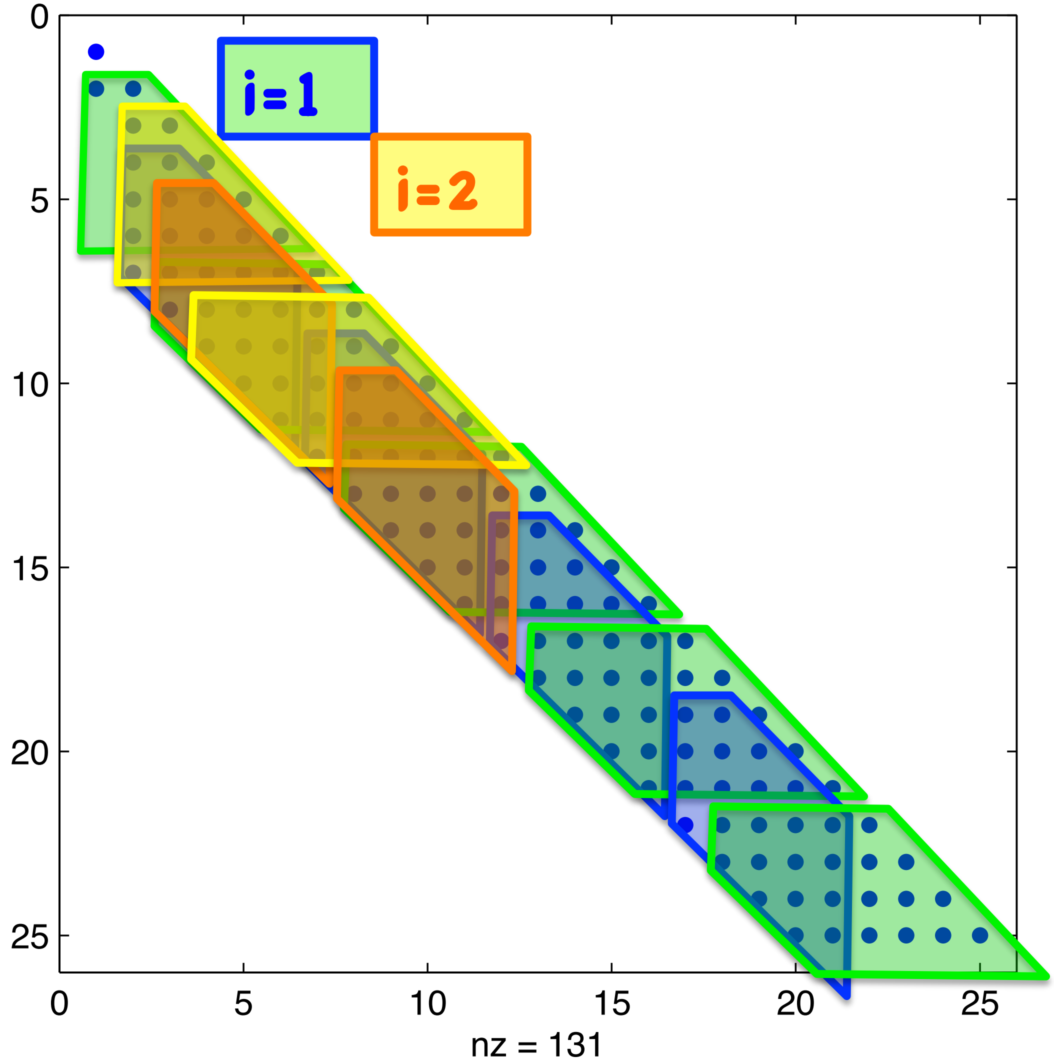
The Bulge chasing algorithm, step -2-



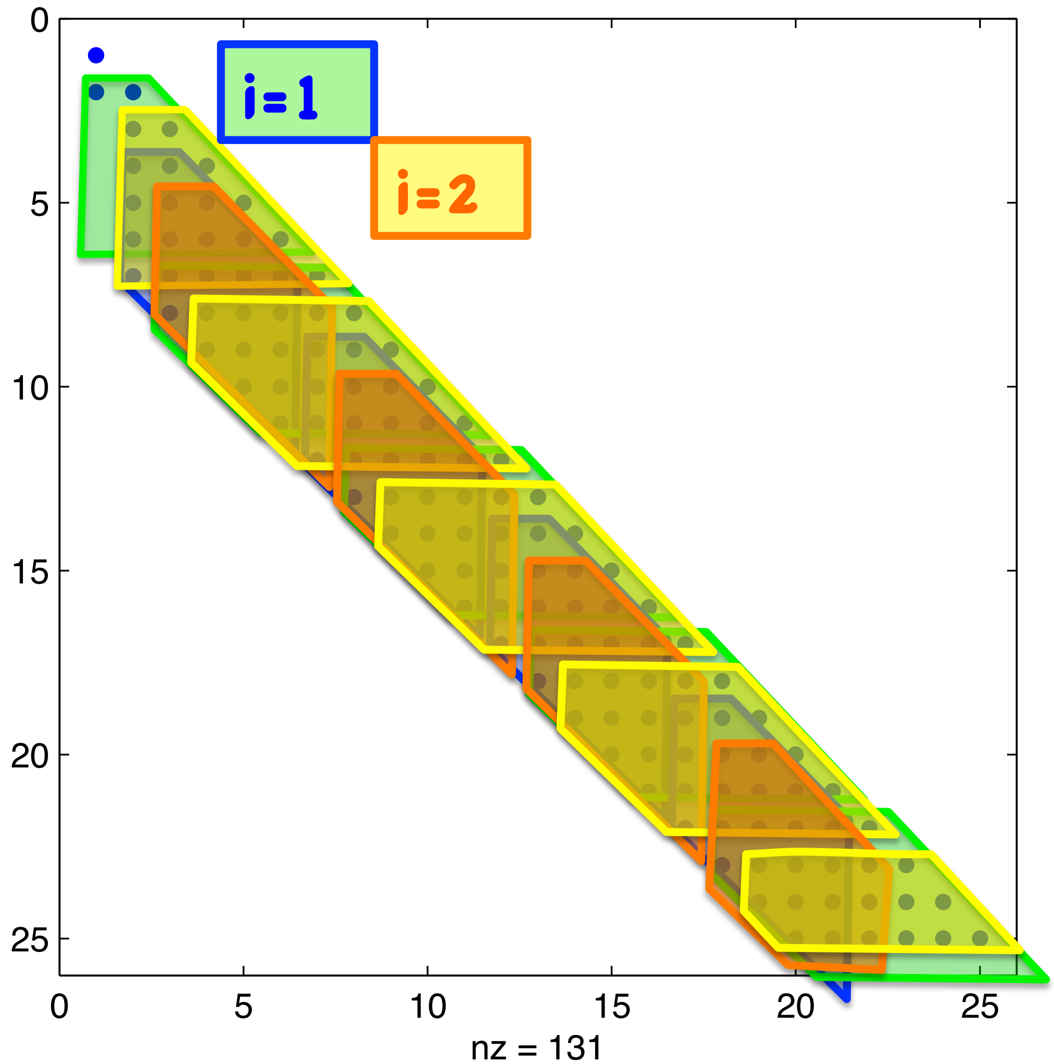
The Bulge chasing algorithm, stage -2-



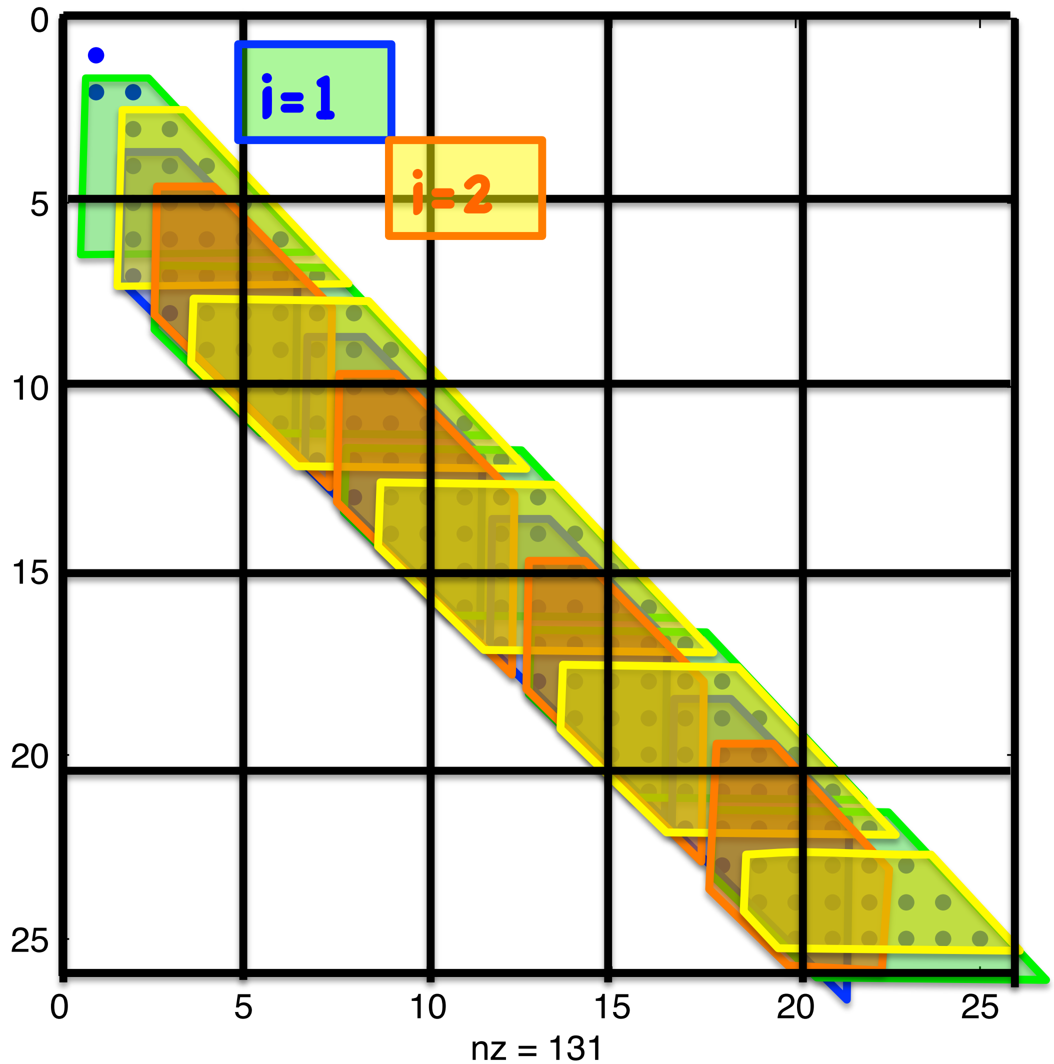
The Bulge chasing algorithm, step -2-



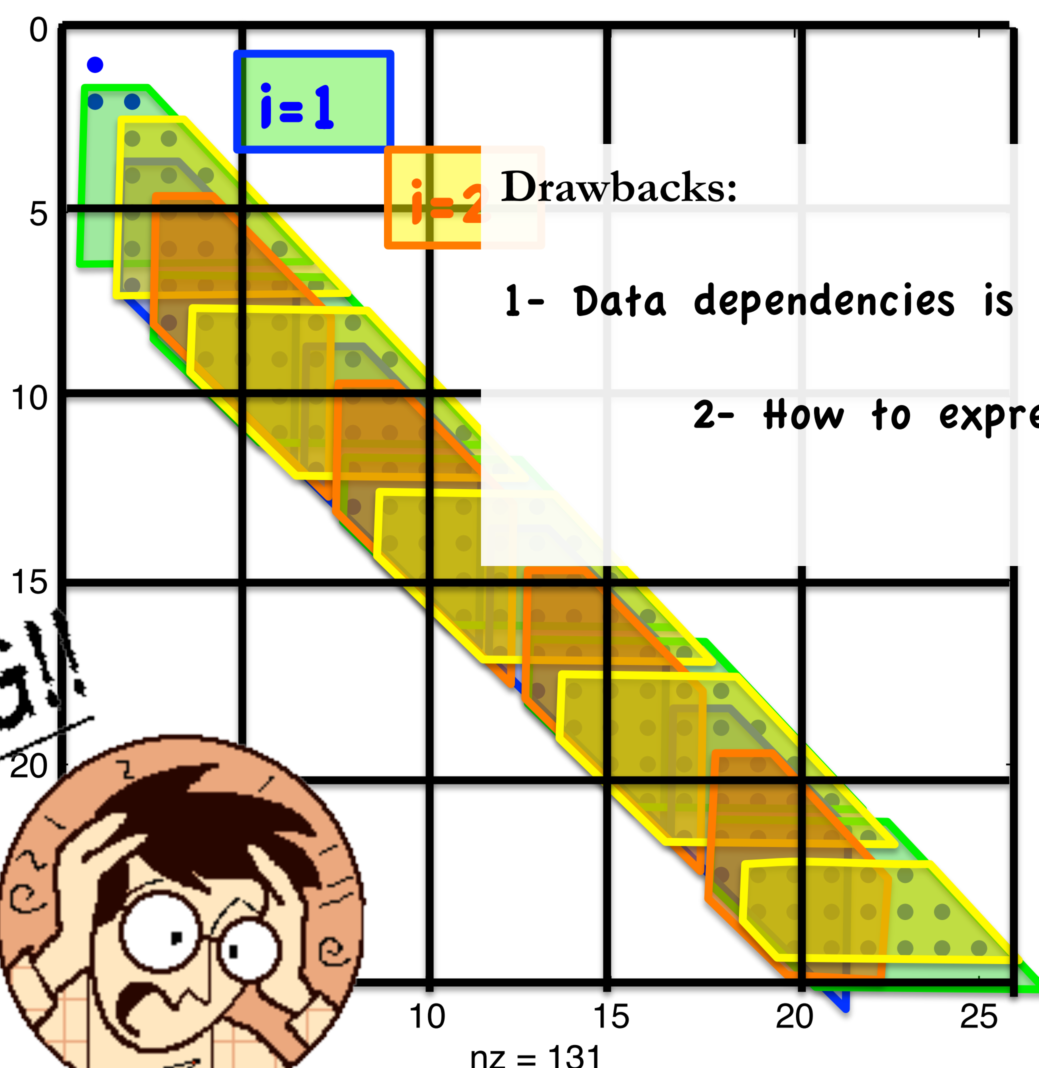
The Bulge chasing algorithm, step -2-



The Bulge chasing algorithm, step -2-



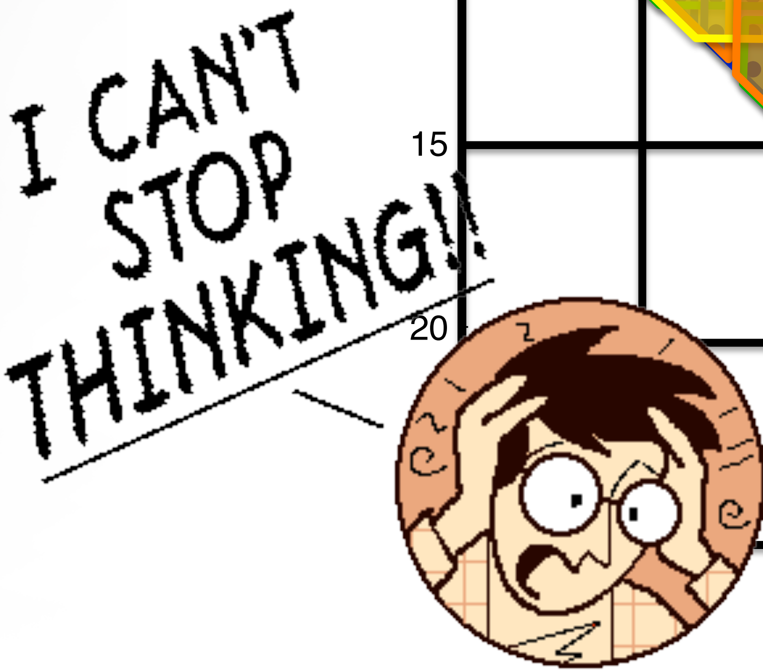
The Bulge chasing algorithm, step -2-



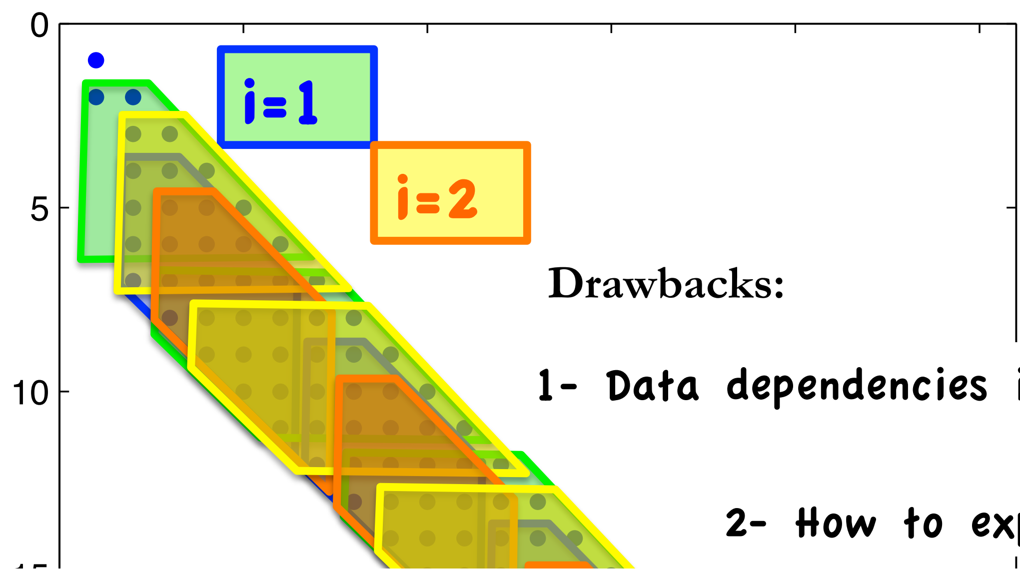
Drawbacks:

1- Data dependencies is very complicated

2- How to express parallelism ?



The Bulge chasing algorithm, step -2-



Drawbacks:

1- Data dependencies is very complicated

2- How to express parallelism ?

Solution:

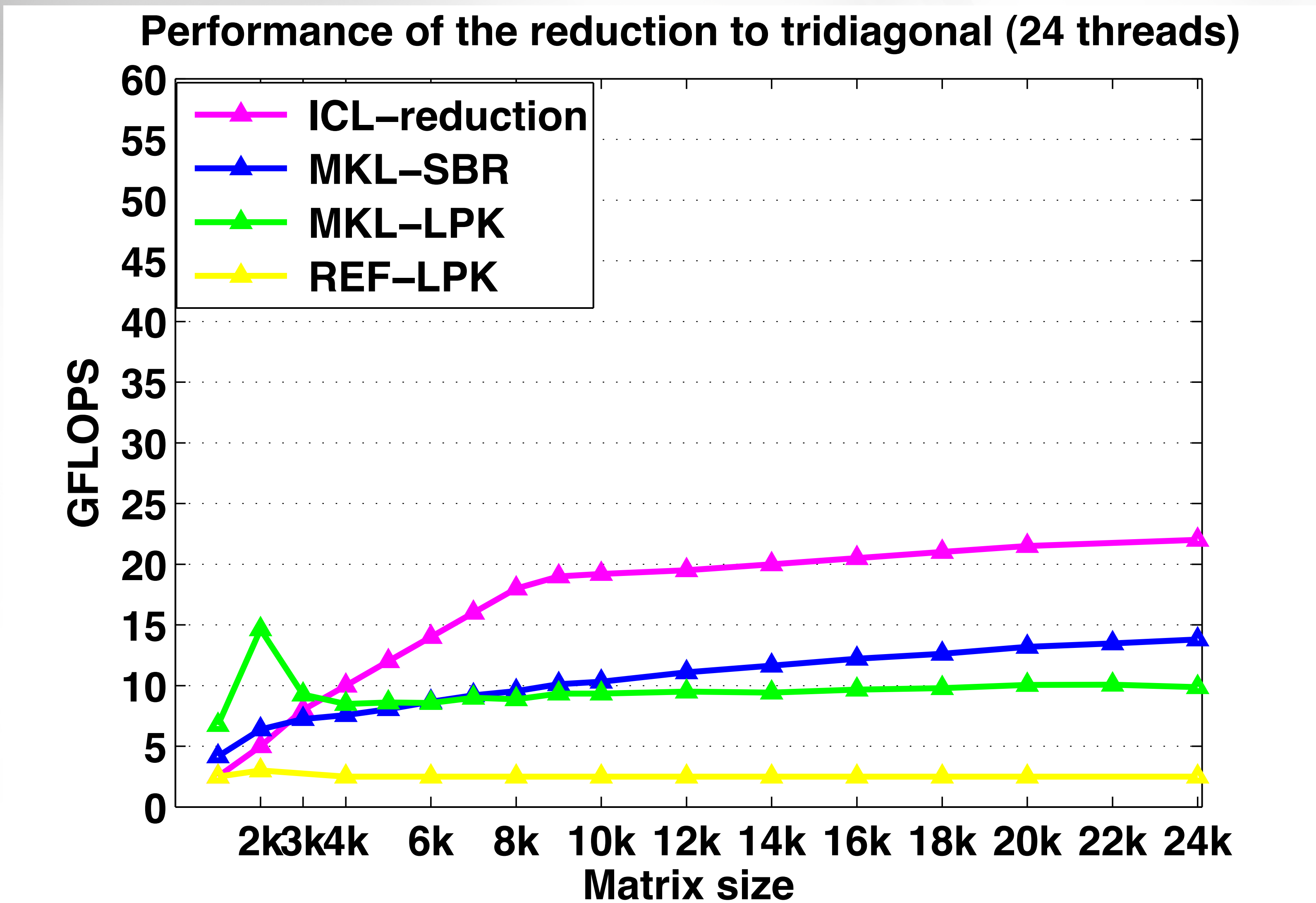
1- Express dependencies a *function-dep* not as *data-dep*

2- Extract a pipelined parallelism

Outline

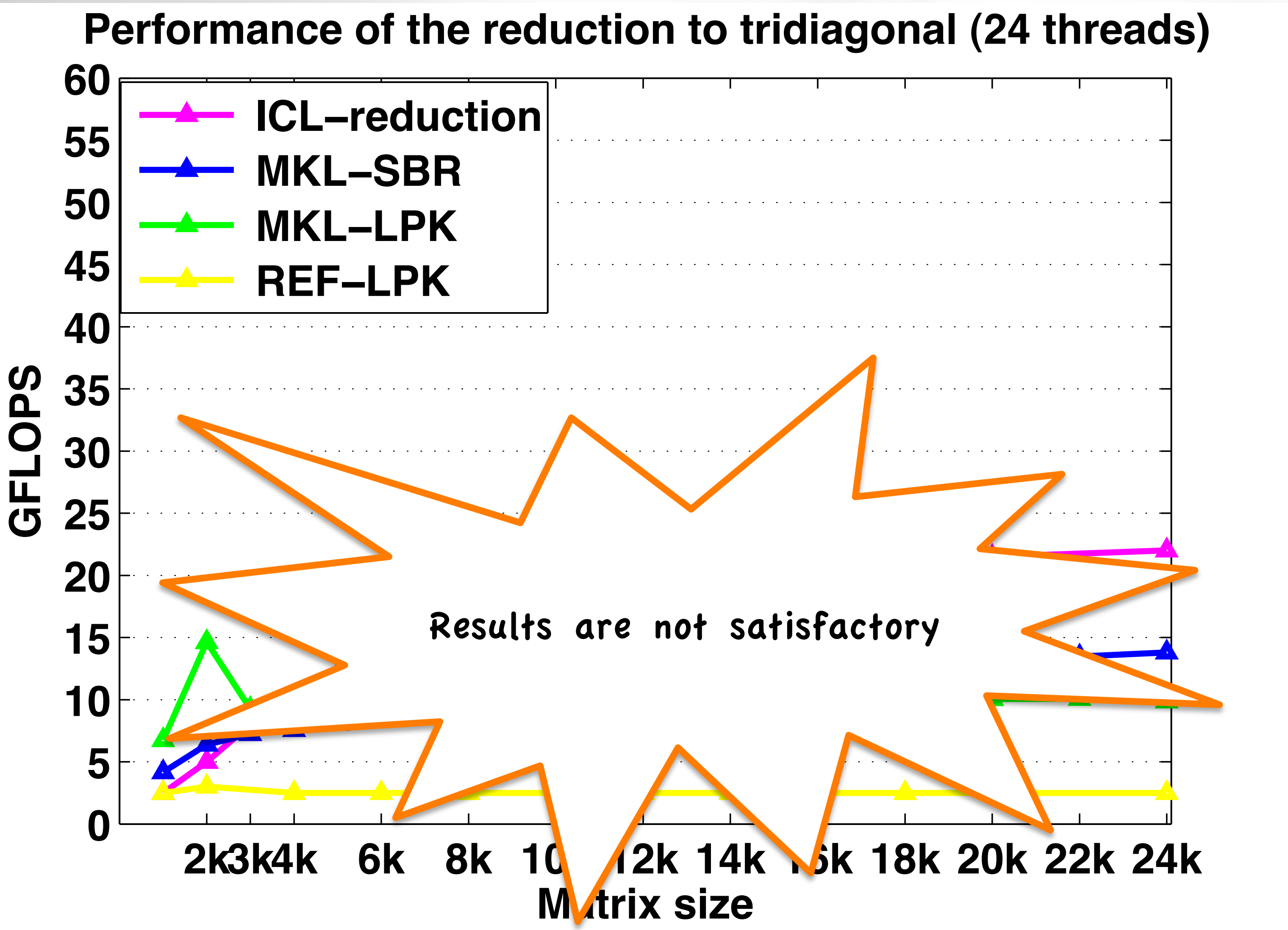
1. Part I : overview
2. a story
3. Part I : The description
4. Part I : The results
5. Part II : advertisement
6. The Burns supper @ cerfacs: some pics
7. I appreciate your contribution and Thanks

The Results of the full reduction



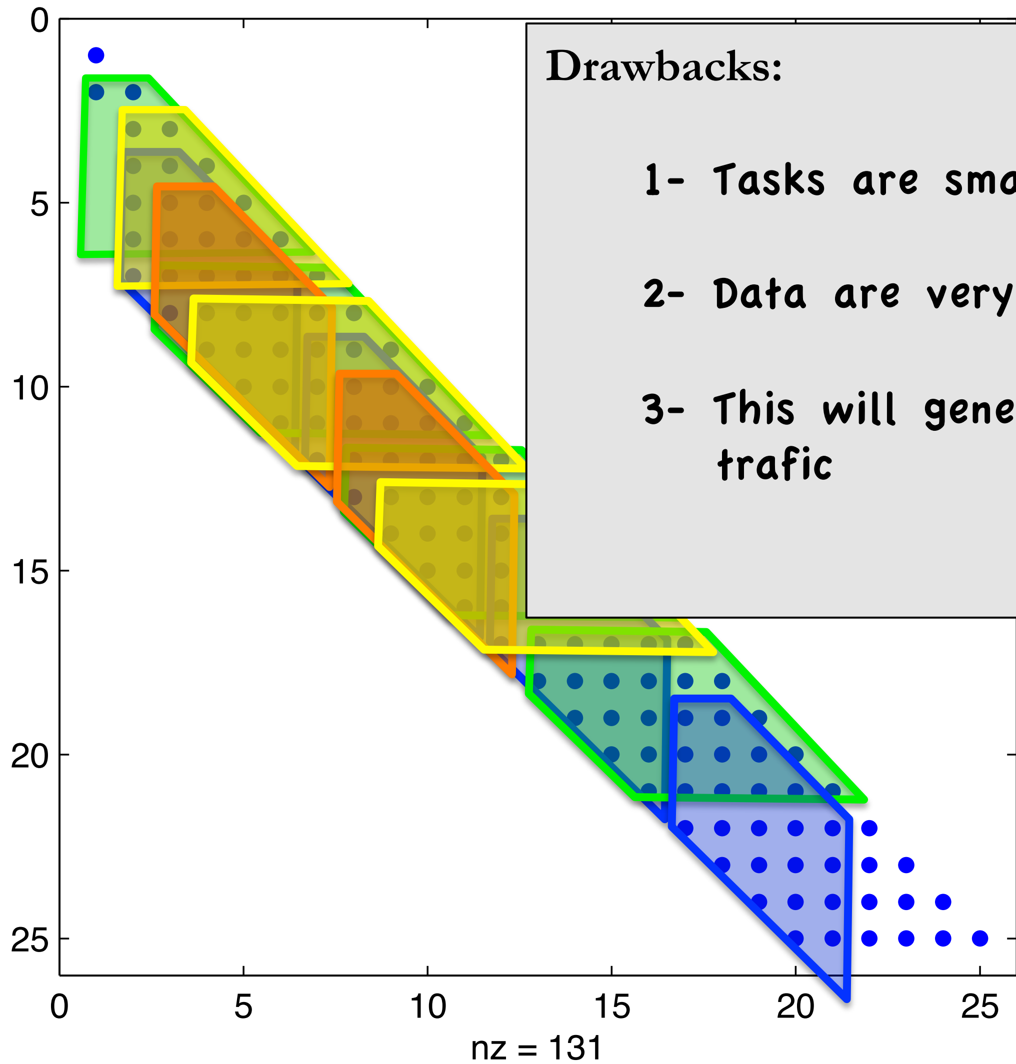
Elapsed time in seconds on eight-socket six-AMD Opteron 2.4 GHz processors with MKL V10.2.4.

The Results of the full reduction



Elapsed time in seconds on eight-socket six-AMD Opteron 2.4 GHz processors with MKL V10.2.4.

The Bulge chasing algorithm, step -2-



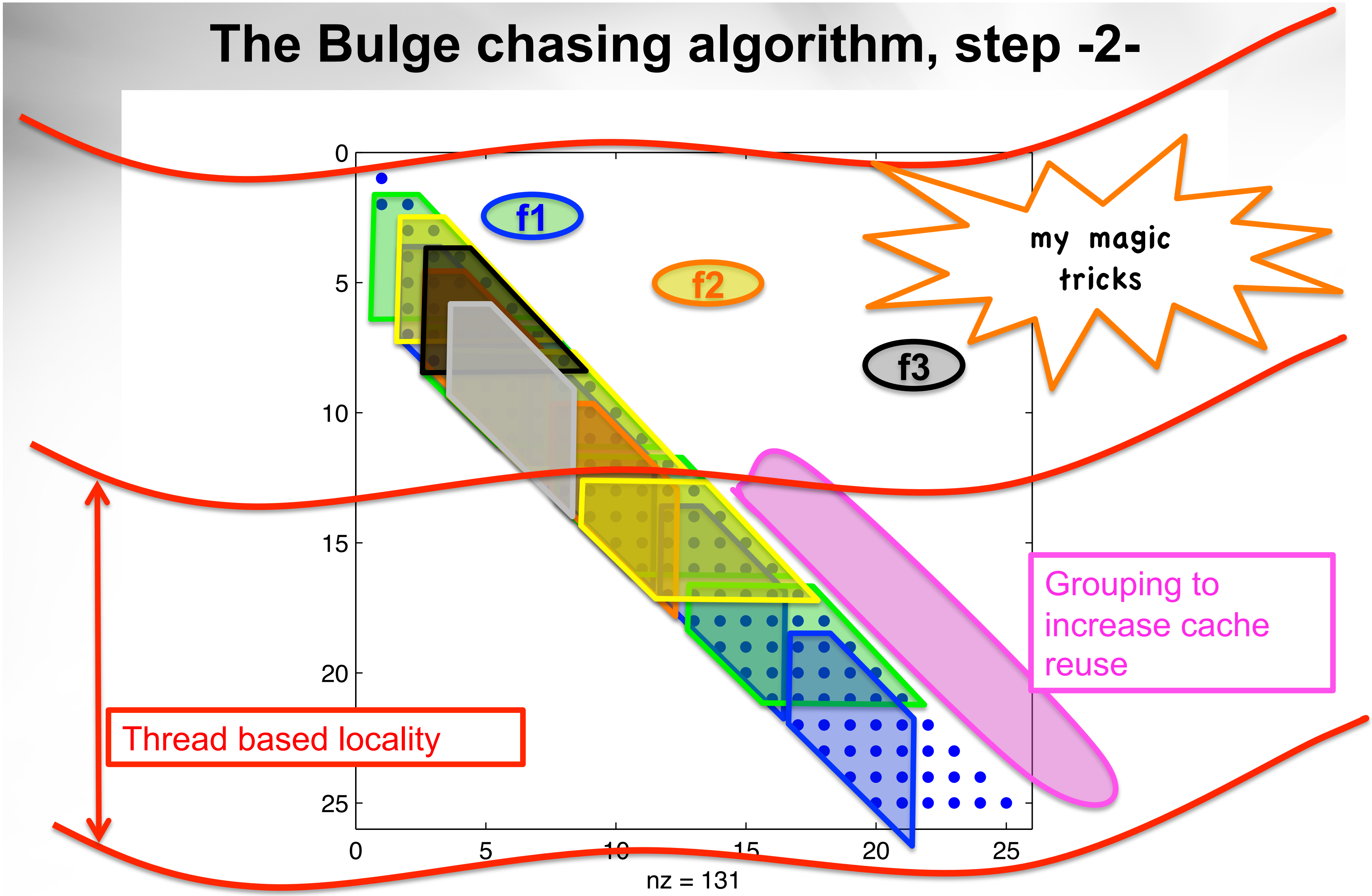
Drawbacks:

1- Tasks are small and fast.

2- Data are very close in memory

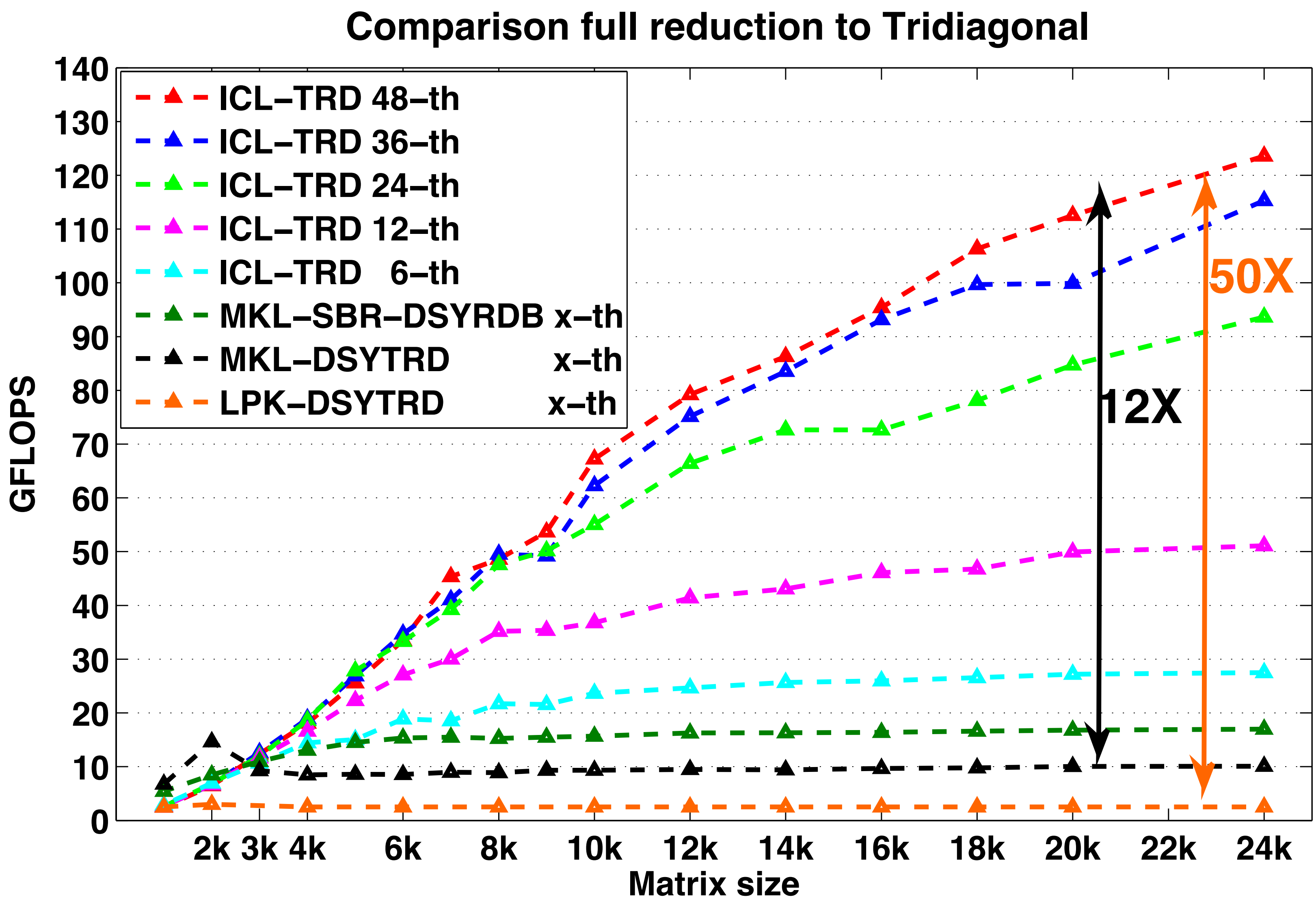
3- This will generate huge memory trafic

The Bulge chasing algorithm, step -2-



The Results

flops formula: $4n^3/3$



Elapsed time in seconds on eight-socket six-AMD Opteron 2.4 GHz processors with MKL V10.2.4.

What about the

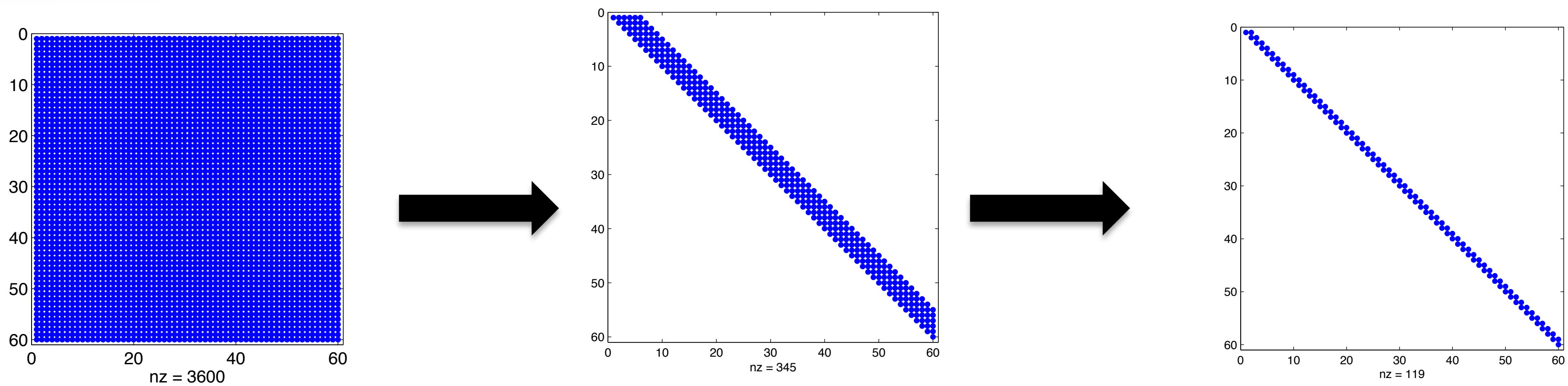
Singular Value Decomposition?

Observations

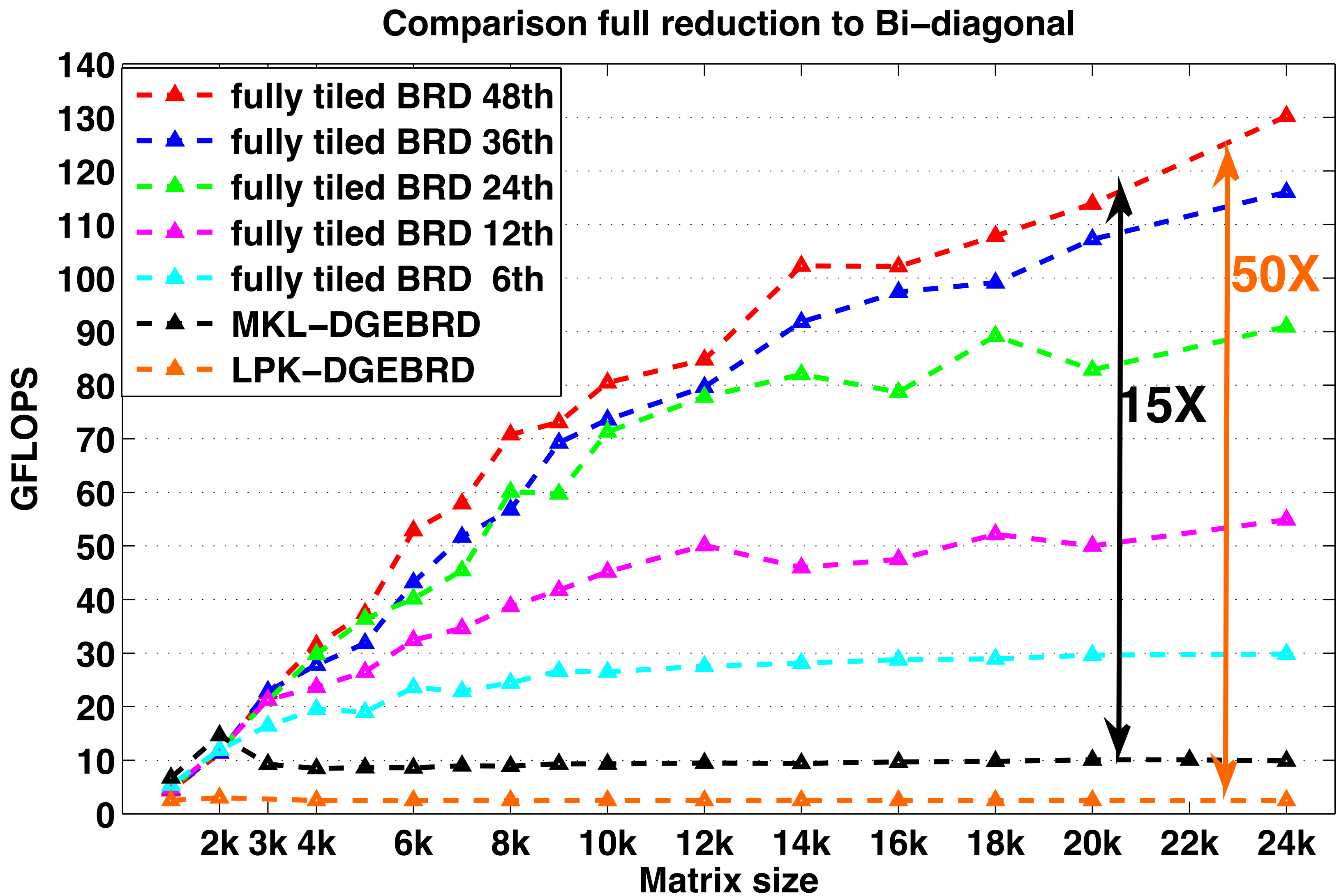
Same concept

1- reduction to upper-band

2- from upper-band to bidiagonal (bulge chasing)



Bulge Chasing- Preliminary results



Elapsed time in seconds on eight-socket six-AMD Opteron 2.4 GHz processors with MKL V10.2.4.

Outline

1. Part I : overview
2. a story
3. Part I : The description
4. Part I : The results
5. Part II : advertisement
6. The Burns supper @ cerfacs: some pics
7. I appreciate your contribution and Thanks

Part II: advertisement

MaPHyS

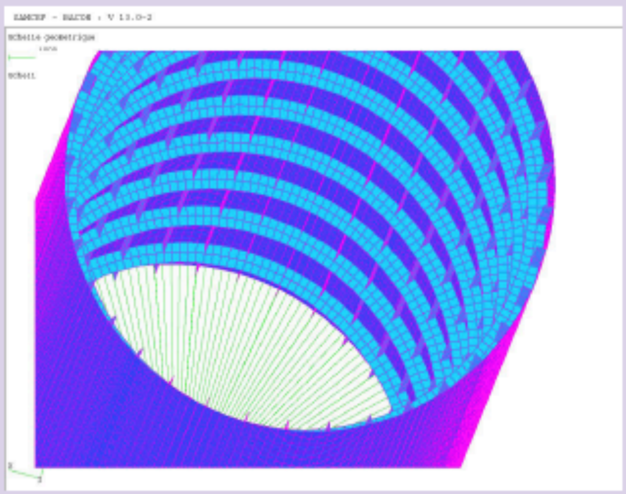
Massively Parallel Hybrid Solver

- I developed MaPHyS during my PhD thesis
- The first beta-version is out distributed by INRIA and CERFACS

Part II: MaPHyS

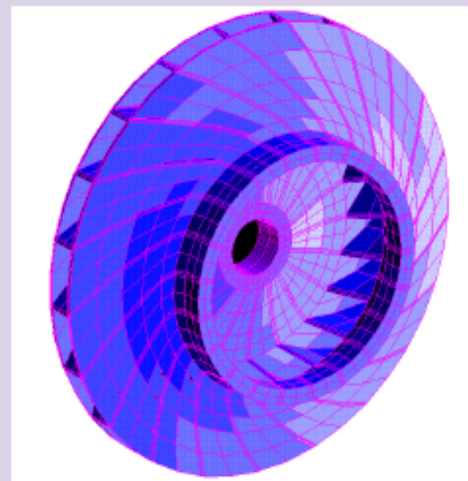
Indefinite systems in structural mechanics S. Pralet, SAMTECH

Fuselage of 6.5 Mdof



- Composed of its skin, stringers and frames
- Midlinn shell elements are used
- Each node has 6 unknowns
- A force perpendicular to the axis is applied

Rouet of 1.3 Mdof

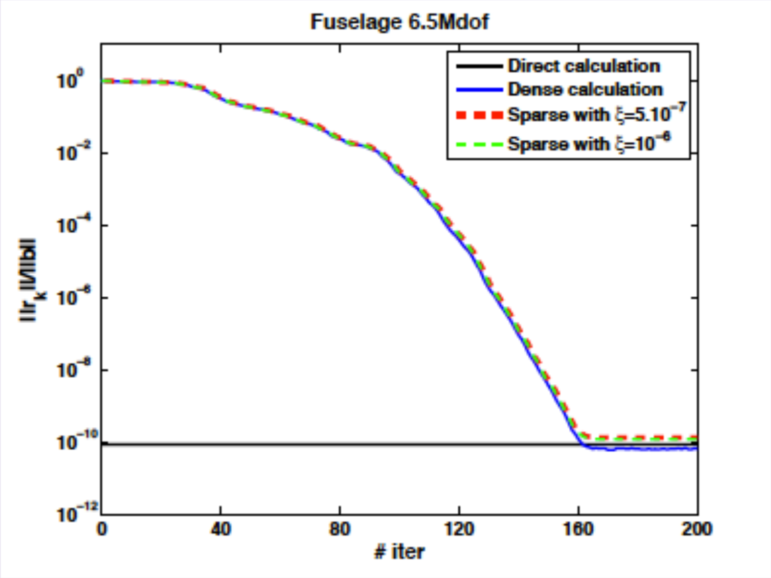


- A 90 degrees sector of an impeller
- It is composed of 3D volume elements
- Cyclic conditions are added using elements with 3 Lagranges multipliers
- Angular velocities are introduced

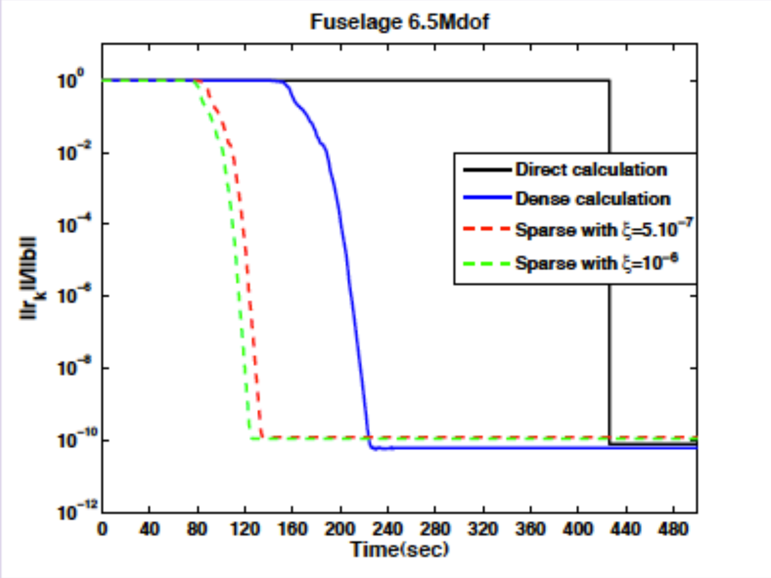
Part II: MaPHyS

MaPHyS: numerical behaviour of the preconditioners

Convergence history



Time history



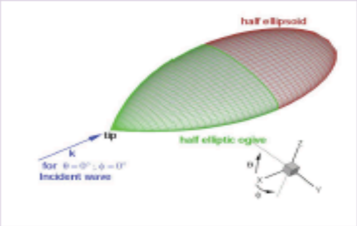
- Fuselage problem of 6.5 Mdof dof mapped on 16 processors
- The sparse preconditioner setup is 4 times faster than the dense one (19.5 v.s. 89 seconds)
- In term of global computing time, the sparse algorithm is about twice faster
- The attainable accuracy of the hybrid solver is comparable to the one computed with the direct solver

Part II: MaPHyS

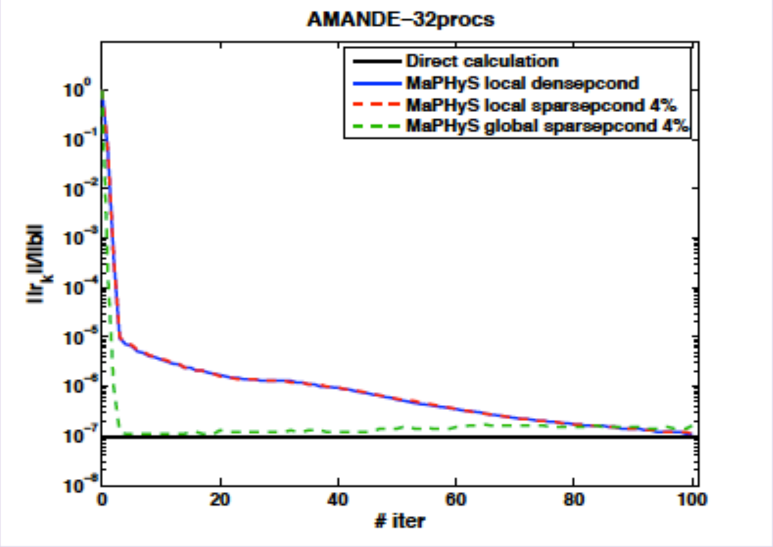
Black-box hybrid solver: problem characteristics

Amande (Almond) problem

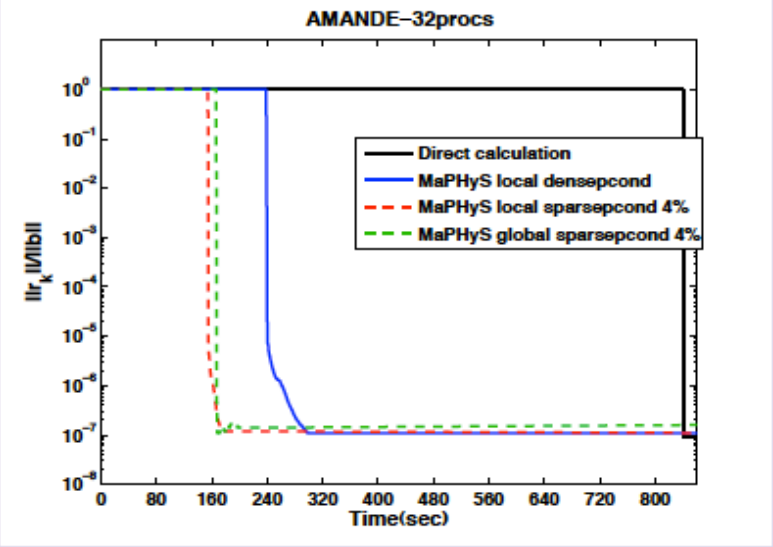
- Electromagnetism problem
- 6,994,683 dof
- 58,477,383 nnz



Convergence history



Time history

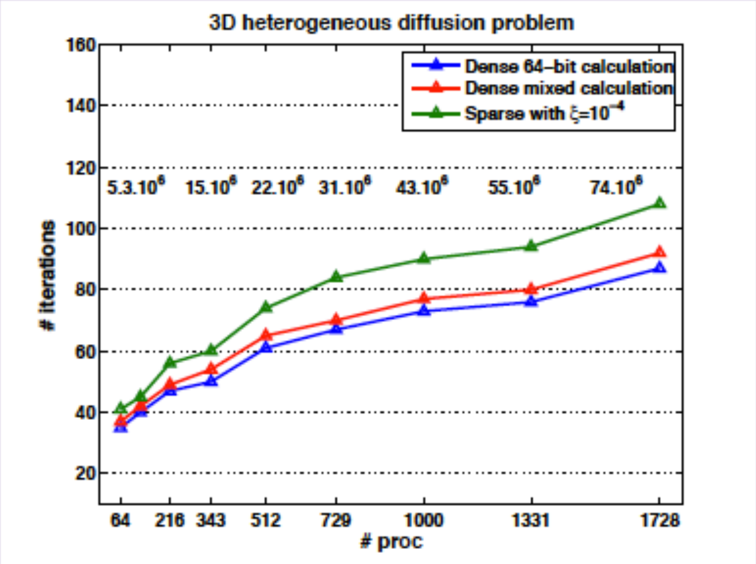


ICL UT

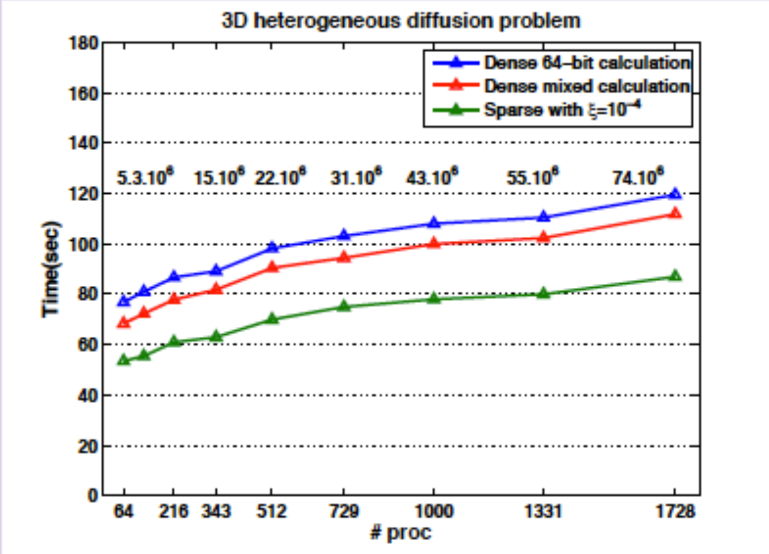
Part II: MaPHyS

Weak scalability on massively parallel platforms

Numerical scalability



Parallel performance



- The solved problem size varies from 2.7 up to 74 Mdof
- Control the grow in the # of iterations by introducing a coarse space correction
- The computing time increases slightly when increasing # sub-domains
- Although the preconditioners do not scale perfectly, the parallel time scalability is acceptable
- The trend is similar for all variants of the preconditioners using CG Krylov solver

ICL UT

Outline

1. Part I : overview
2. a story
3. Part I : The description
4. Part I : The results
5. Part II : advertisement
6. The Burns supper @ cerfacs: some pics
7. I appreciate your contribution and Thanks

Part III: the Burns supper @ Cerfacs

Part III: the Burns supper @ Cerfacs

The end

I appreciate your attention,

Thank you very much