Milovy, August 2002

COMBINING MULTILEVEL PRECONDITIONERS WITH DOMAIN DECOMPOSITION

Gérard MEURANT

CEA/DIF

Joint work with Pierre Leca (CEA/DIF)

Problem

We would like to efficiently solve linear systems arising from PDEs discretization (FD or FE) on Terascale parallel computers

$$Ax = b$$

A sparse symmetric positive definite of order n

The iterative method must be (almost) **scalable**

We will use PCG. The preconditioner must be such that:

• the number of iterations is (almost) constant, when the problem size is increased (the condition number must be independent of h for PDEs or n)

• the complexity of applying the preconditioner is proportional to n

• easy to construct and use on a parallel computer

Three known possibilities:

◦ multilevel (algebraic multigrid–like) methods

◦ domain decomposition (DD)

◦ combination of both

Today we concentrate on the first one, but use DD techniques for parallelization

Multilevel preconditioners

◦ Algebraic methods (grid $\equiv$ (sub) set of unknowns without overlapping)

◦ Algebraic MultiGrid (AMG)–like V–cycle:

Starting from the zero vector:

0– if we are on the coarsest level, solve exactly by Gaussian elimination, otherwise

1– do ν iterations of smoothing

2– restrict the residual r to $r_c = Rr$

3– recursively solve $A_c e_c = r_c$, $A_c = RAP$, $R = P^T$

4– interpolate e_c to $e = Pe_c$

5– add the correction e to the current iterate

6– do ν iterations of smoothing

Generally, we will use $\nu = 1$. We have to define:

◦ the smoother

◦ how to construct the coarse grids (coarsening alg.)

◦ the interpolation

There are many choices!

We want to illustrate 2 points:

1) full parallelization is not easy

Sometimes, introducing more parallelism leads to the loss of scalability

2) it is sometimes better to partition the graph of the influence matrix rather than the graph of A

Smoothers

◦ Symmetric Gauss–Seidel (not //, needs coloring of the nodes)

◦ Incomplete Cholesky (not // either) $M = LDL^T$

- IC(0) no fill–in
- IC with some fill–in on values
- IC with some fill–in on levels of fill

$$LD^{-1}L^T(x^{k+1} - x^k) = b - Ax^k$$

We'll see later on how to introduce parallelism

◦ Least squares polynomial

$$\int_a^b (1 - \lambda q(\lambda))^2 w(\lambda)\, d\lambda, \quad q \in \mathcal{Q}_k,$$

a and b Gerschgorin bounds for the eigenvalues

It is usually more efficient to use a low order (1 or 2) polynomial. This one is // (matrice×vector)

◦ Approximate inverse AINV from M. Benzi and al. $M = ZD^{-1}Z^T$ where the matrix Z is upper triangular with $1's$ on the diagonal and D is diagonal

Approximate A–orthogonalization:

$Z = I$

$d_1 = a_{1,1}$

for $i = 2, \ldots, n$

drop the entries $z_{k,i}$ such that $|z_{k,i}| \leq \tau \|a_i\|_\infty$

for $j = i, \ldots, n$

$d_j = a_i^T z_j$

end

for $j = i + 1, \ldots, n$

$z_j = z_j - \frac{d_j}{d_i} z_i$

end

end

The parameter τ defines which entries of Z are kept

Works fine for H–matrices, for general SPD matrices use SAINV (Stabilized AINV).

Smoother: Richardson iteration defined as (matrix $\times$)

$$x^{k+1} = x^k + M(b - Ax^k)$$

The use of M is parallel but the construction of Z is not // (recurrences)

Other choice: SPAI from Huckle and Grote

$$min\|I - AM\|_F$$

Computation of M is parallel (columns of M are independent)

Problem: symmetry and positive definiteness of M

The influence matrix

$$\mathcal{N} = \{1, \ldots, n\}$$

$$\mathcal{N} = F \cup C$$

Influence matrix: standard AMG choice for M–matrices

$$S_i = \{j | -a_{i,j} > \theta \max_{k \neq i}(-a_{i,k}), \quad \theta < 1\}$$

This gives the rows of S (by padding with zeros and 1 if in S_i)

Generalization:

$$S_i^A = \{j \mid |a_{i,j}| > \tau \max_{k \neq i} |a_{i,k}|, \quad \tau < 1\}$$

This choice is denoted by 'a'. Keep at least a 1 for the largest element in magnitude ('b').

The coarsening algorithm

The "standard"('st') algorithm is:

Weights w_i = the number of points that depend on i (using S)

1- choose the first point i with maximal weight as a C point

2- assign the points that i influences (using S) as F points

3- increment by 1 the weights of the points influencing these new F points (to give more chances to be selected as C points)

4- decrease by 1 the weights of points that depends on i

repeat from step 1- until all points are labeled

◦ There are many other choices for the coarsening algorithm

◦ Remark that if we change the interpolation algorithm the number and location of coarse nodes change (on coarse levels)

◦ This is not parallel (update of the weights)

The interpolation algorithm

◦ Standard AMG ('st') uses $Ae = 0$, $i \in F, j \in C$

$$\omega_{i,j} = -\frac{a_{i,j} + \sum_{k \in D_i^S} \frac{a_{i,k} a_{k,j}}{\sum_{m \epsilon C_i} a_{k,m}}}{a_{i,i} + \sum_{k \in D_i^W} a_{i,k}}$$

This uses $e_j \approx e_i$ for weak connections and a weighted average for F connections

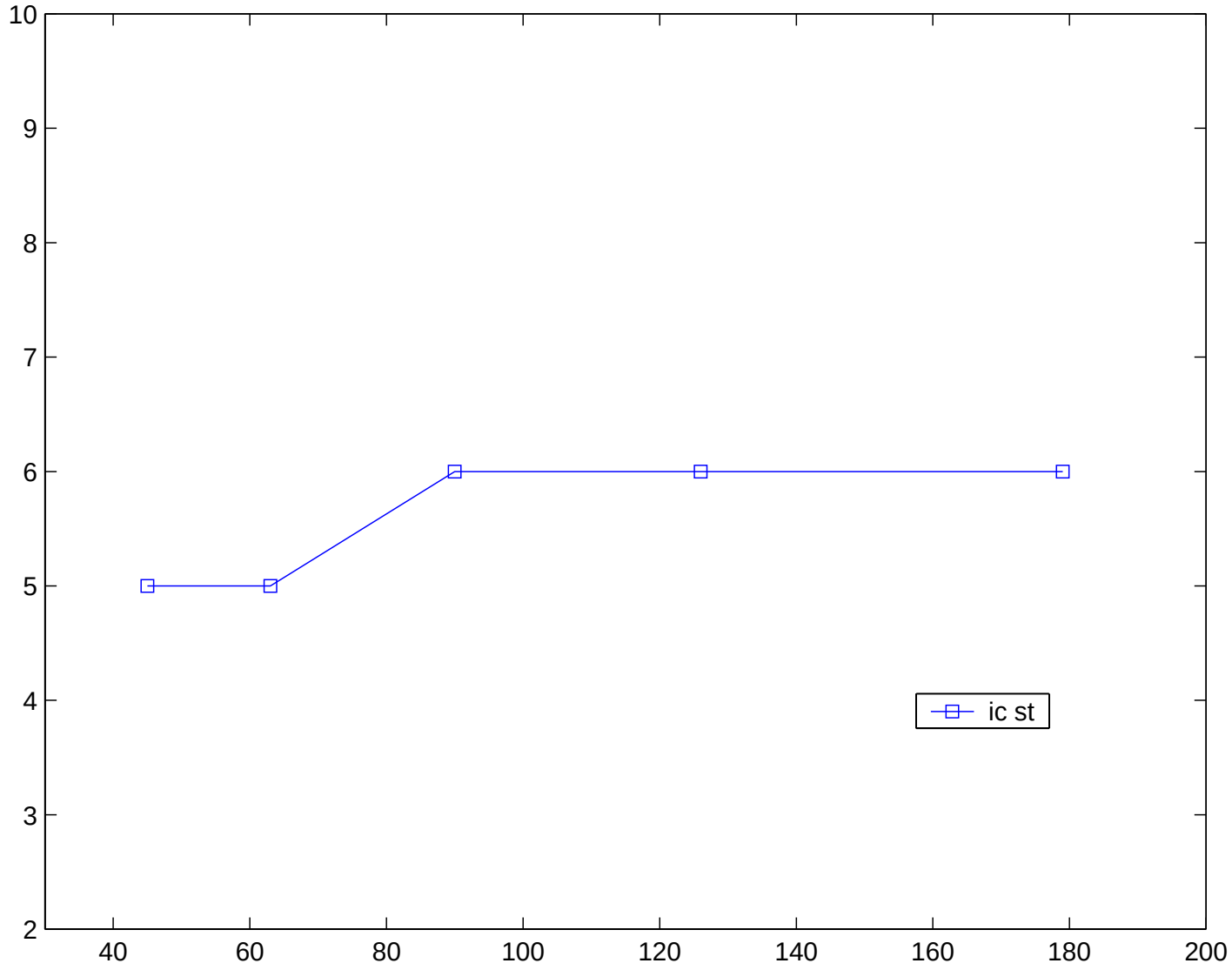
Note that a given F point needs to have at least one coarse point in its neighborhood in the graph of A

The coarse matrices

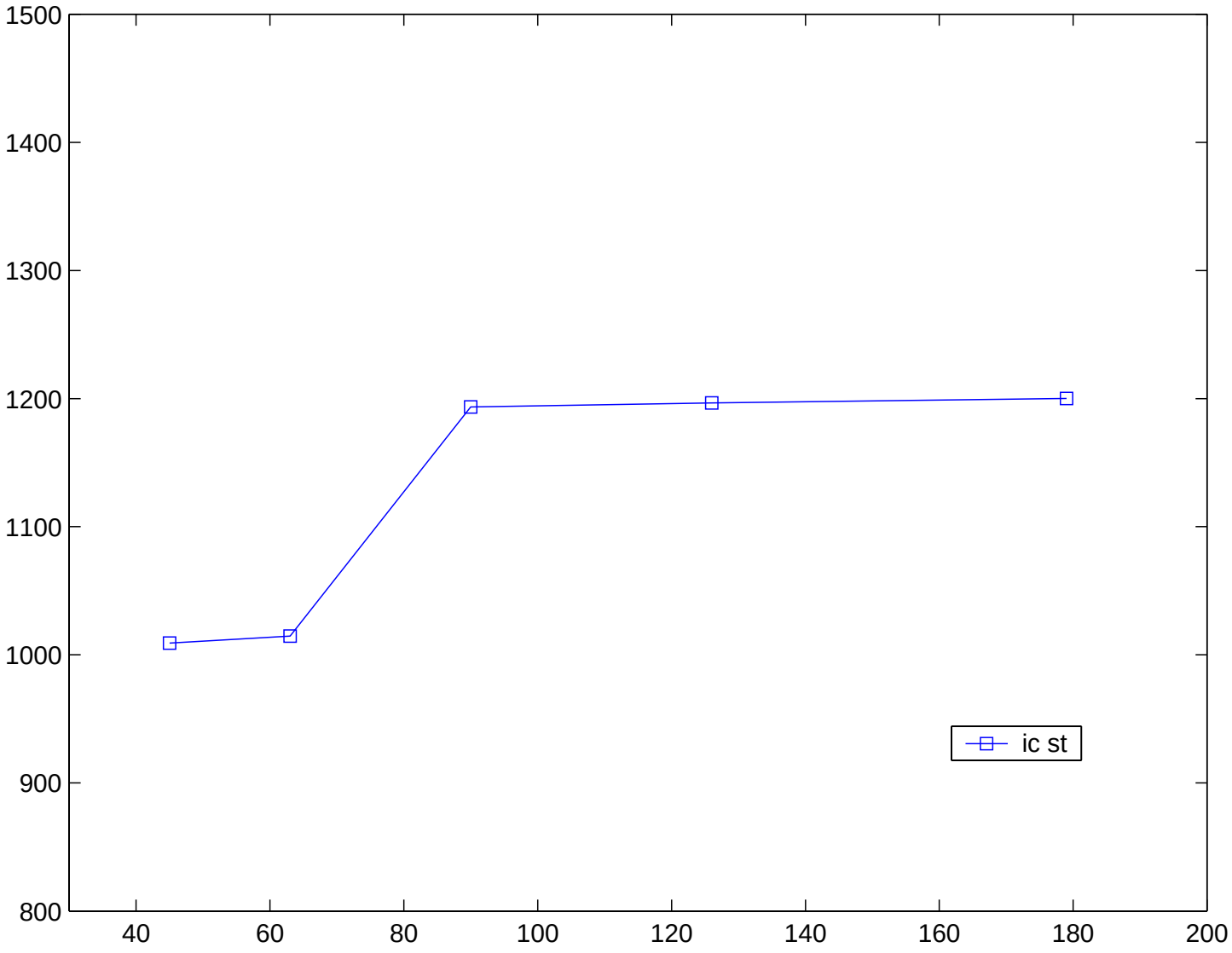
The interpolation algorithm defines P and $R = P^T$

$$A_C = RAP$$

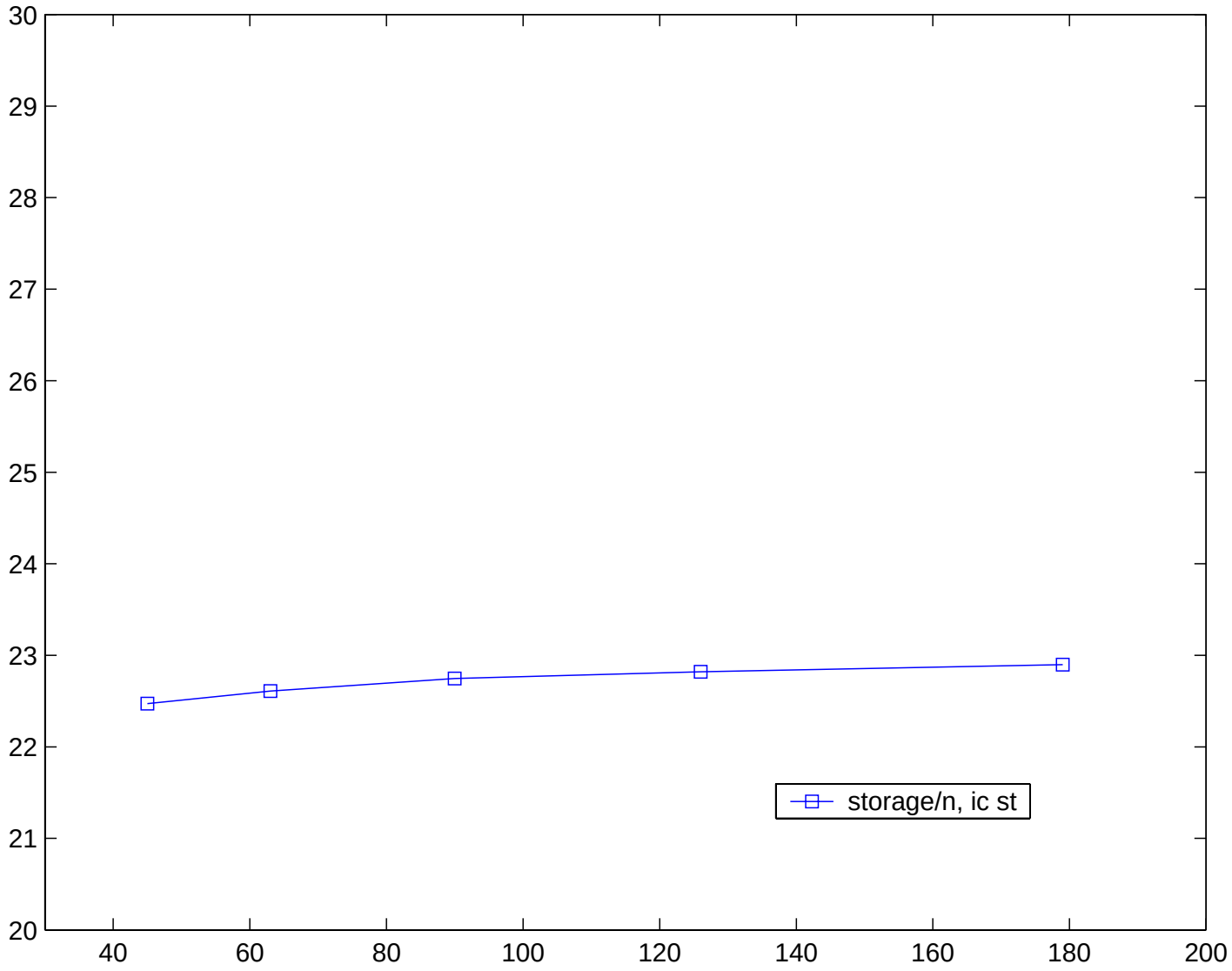
The sequential versions of the smoothers and coarsening algorithm are scalable (at least for the Poisson problem)



scalability for the Poisson equation, $m \times m$ mesh



scalability for the Poisson equation, $m \times m$ mesh



scalability for the Poisson equation, $m \times m$ mesh

• How to parallelize the construction of the smoother (IC or AINV)?

◦ partition the domain into subdomains and interfaces or with overlapping

• In fact, we partition the graph of A (or the graph of S)

◦ with interfaces we order the subgraphs first, then the interfaces

◦ In IC or AINV ignore the fill across different subgraphs, but keep the fill inside subgraphs or with the interfaces

◦ With overlapping we compute the factorizations including the overlapping regions and restrict

This allows to factor the subgraphs independently. Z or L are block diagonal (up to a permutation) except for the interface columns (or rows) when they exist.

The solve with AINV is fully parallel. The solve with IC needs a synchronization before solving for the interfaces (less parallel)

• However, with overlapping positive definiteness is not guaranteed

How do we partition the graph into subgraphs and interfaces?

◦ Use modifications of Gilbert's Matlab package

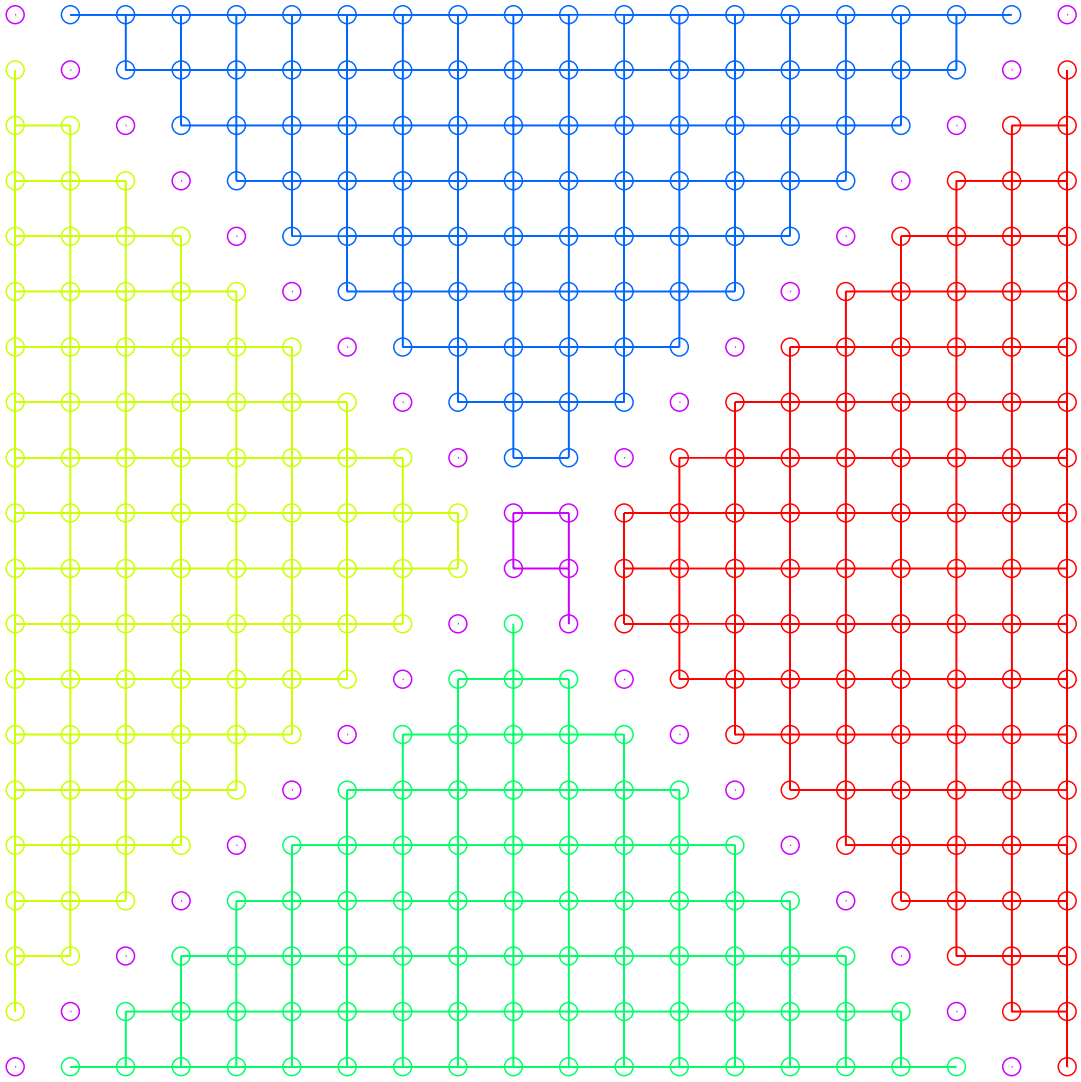
Three available methods (recursive bisection)

- Reverse Cuthill McKee (RCM)
- Spectral
- Geometric

Results do not depend much on the partitioning algorithm

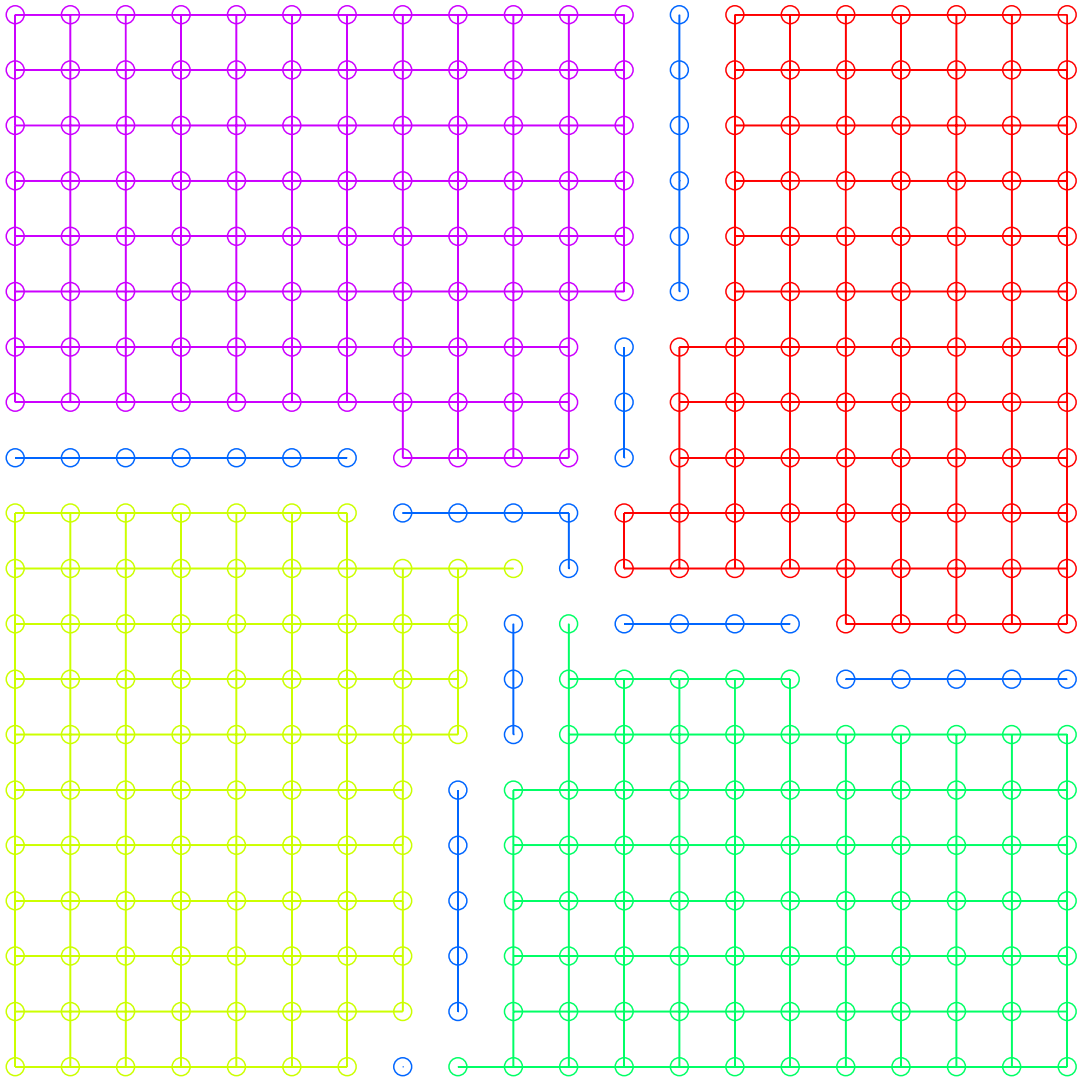
We partition only the finest level graph. Could have a load balancing problem on coarser levels

Examples with PDE problems



143 cut edges

RCM partitioning of the Poisson equation matrix



92 cut edges

Spectral partitioning of the Poisson equation matrix

Parallel coarsening algorithms

◦ The standard algorithm is sequential

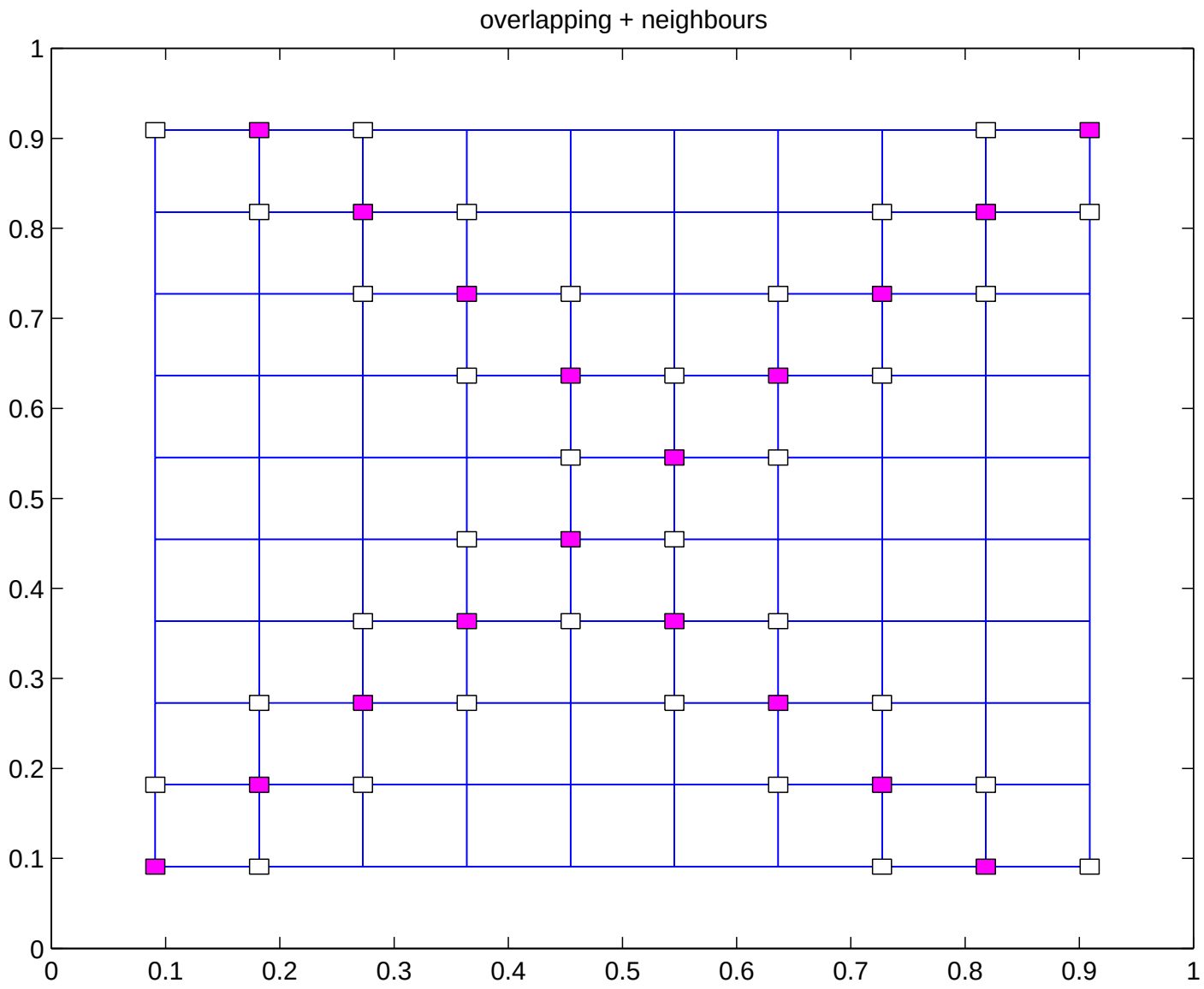
◦ With overlapping using 'st' on each subdomain does not work well

• Start by coarsening the overlapping

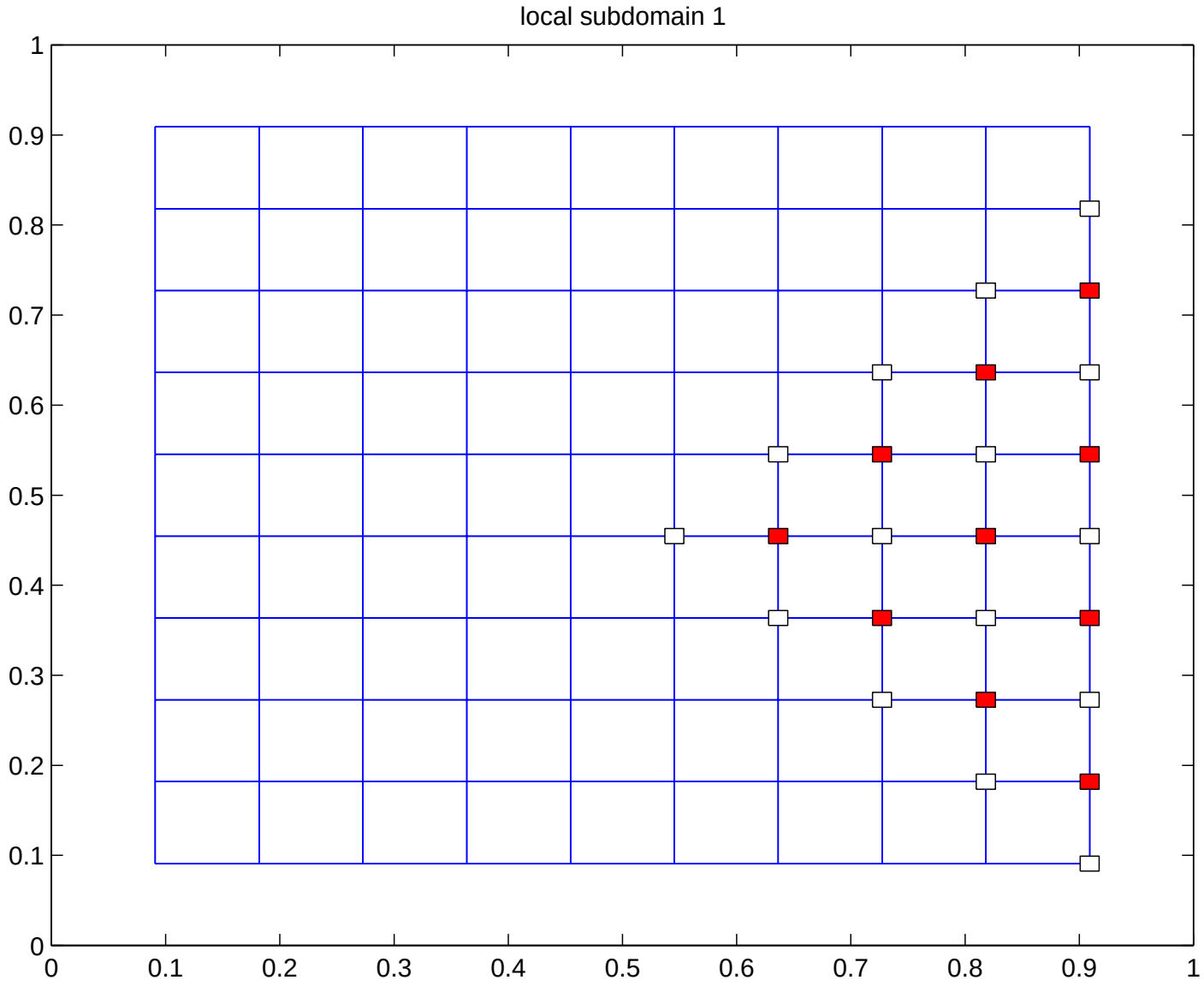
• Flag the neighbours of C points as F points (without $F - F$ connections)

• Coarsen the subdomains in parallel

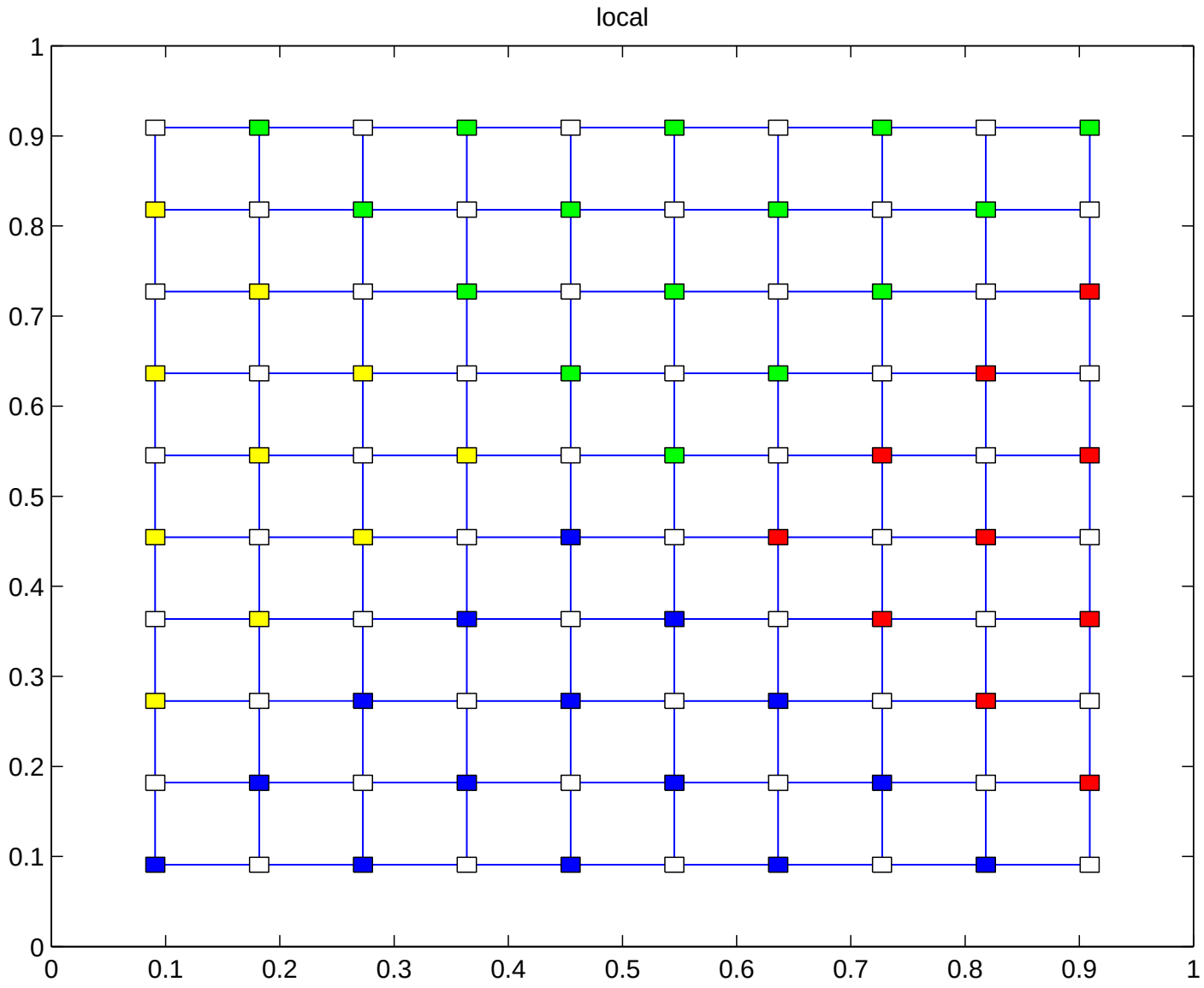
There exists another parallel algorithm by Cleary, Falgout, Henson and Jones



Coarsening the overlapping for the Poisson problem, four subdomains



First subdomain for the Poisson problem, four subdomains



Global result of the coarsening for the Poisson problem, four subdomains

Numerical experiments

5 point finite differences, unit square, $m \times m$ mesh

b random

$x^0 = 0$

stopping criterion $\|r^k\| \leq 10^{-10}\|r^0\|$

Done on my PC (Sony XG9, Intel 500 Mhz) using Matlab (no parallelism so far and "small" problems!)

◦ Poisson equation

◦ Anisotropic problem

diffusion coeff=1 in x and 100 in y

PCG for Poisson equation, $m = 40$, $\tau = 0.05$

multilevel, ('id', 'b', 'sd', 'st'), $F - F$ connections

nb sd	nb it	flops	storage
1	5	1 598 253	35 550
2	14	4 040 899	35 562
4	14	4 107 883	36 351
8	14	4 182 730	37 247
16	16	4 763 745	36 956
32	16	5 029 569	34 040

PCG for Poisson equation, $m = 40$, $\tau = 0.05$

multilevel, ('id', 'b', 'sd', 'st'), refuse $F - F$ connections on fine level

nb sd	nb it	flops	storage
1	5	1 598 253	35 550
2	8	2 484 338	36 790
4	7	2 186 602	36 555
8	8	2 517 864	37 529
16	9	2 824 986	37 545
32	11	3 586 071	34 670

PCG for Poisson equation, $m = 40$, $\tau = 0.05$

multilevel, ('id', 'b', 'sd', 'st'), coarse interface

nb sd	nb it	flops	storage
1	5	1 598 253	35 550
2	7	2 384 306	40 327
4	7	2 482 349	41 528
8	6	2 222 733	41 562
16	6	2 086 997	36 673
32	6	2 251 702	38 653

PCG for Poisson equation, $m = 40$, $\tau = 0.05$

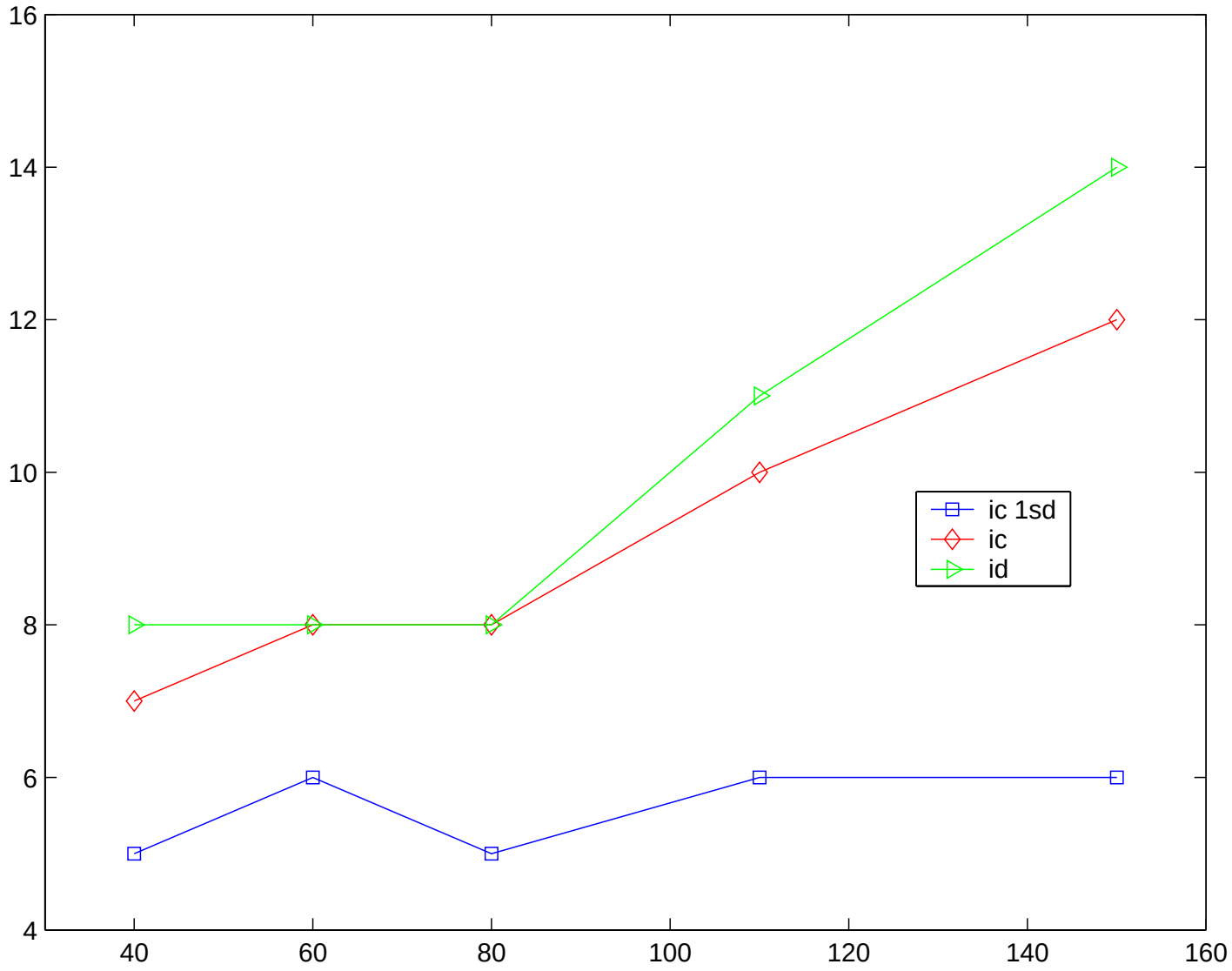
multilevel, ('ad', 'b', 'sd', 'st'), refuse $F - F$ connections on fine level

nb sd	nb it	flops	storage
1	14	3 929 793	36 005
2	15	4 277 693	36 882
4	14	3 971 013	36 611
8	13	3 749 853	37 236
16	12	3 502 164	37 133
32	12	3 750 659	34 536

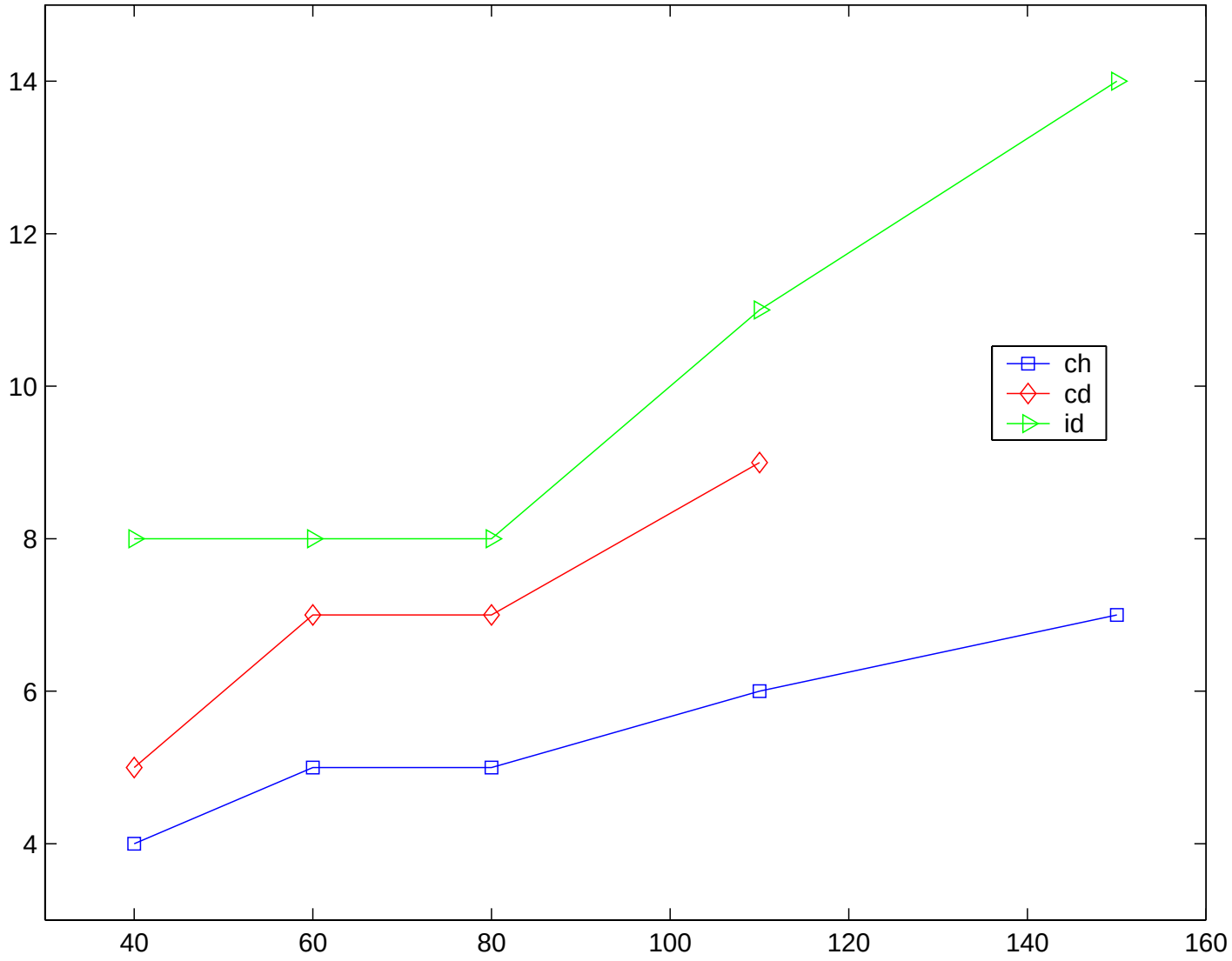
PCG for Poisson equation, $m = 40$, $\tau = 0.05$

multilevel, ('po', 'b', 'sd', 'st'), refuse $F - F$ connections on fine level

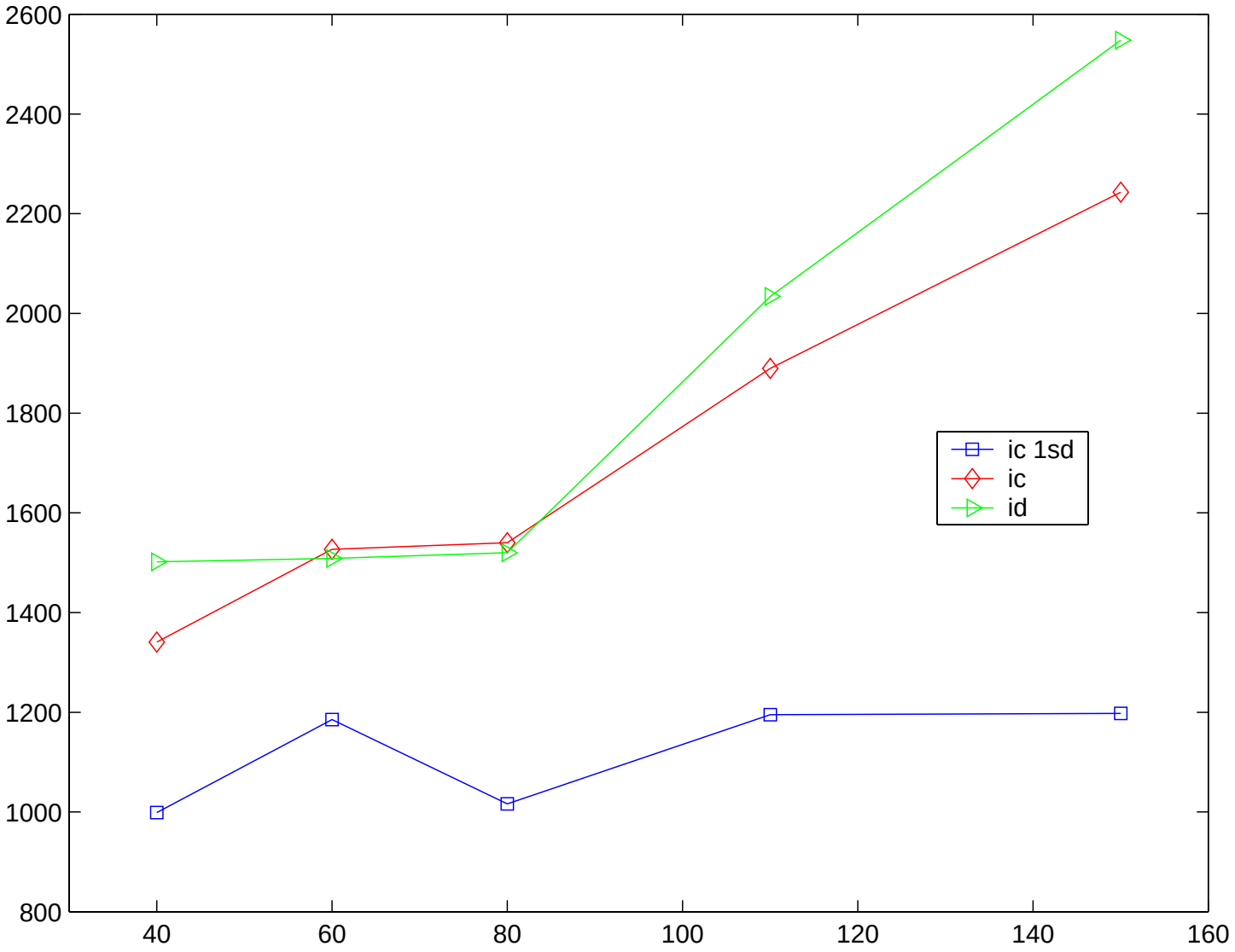
nb sd	nb it	flops	storage
1	7	2 852 717	23 128
2	11	4 442 625	23 993
4	9	3 658 013	23 913
8	10	4 129 533	24 665
16	11	4 507 857	24 979
32	12	4 924 195	23 622



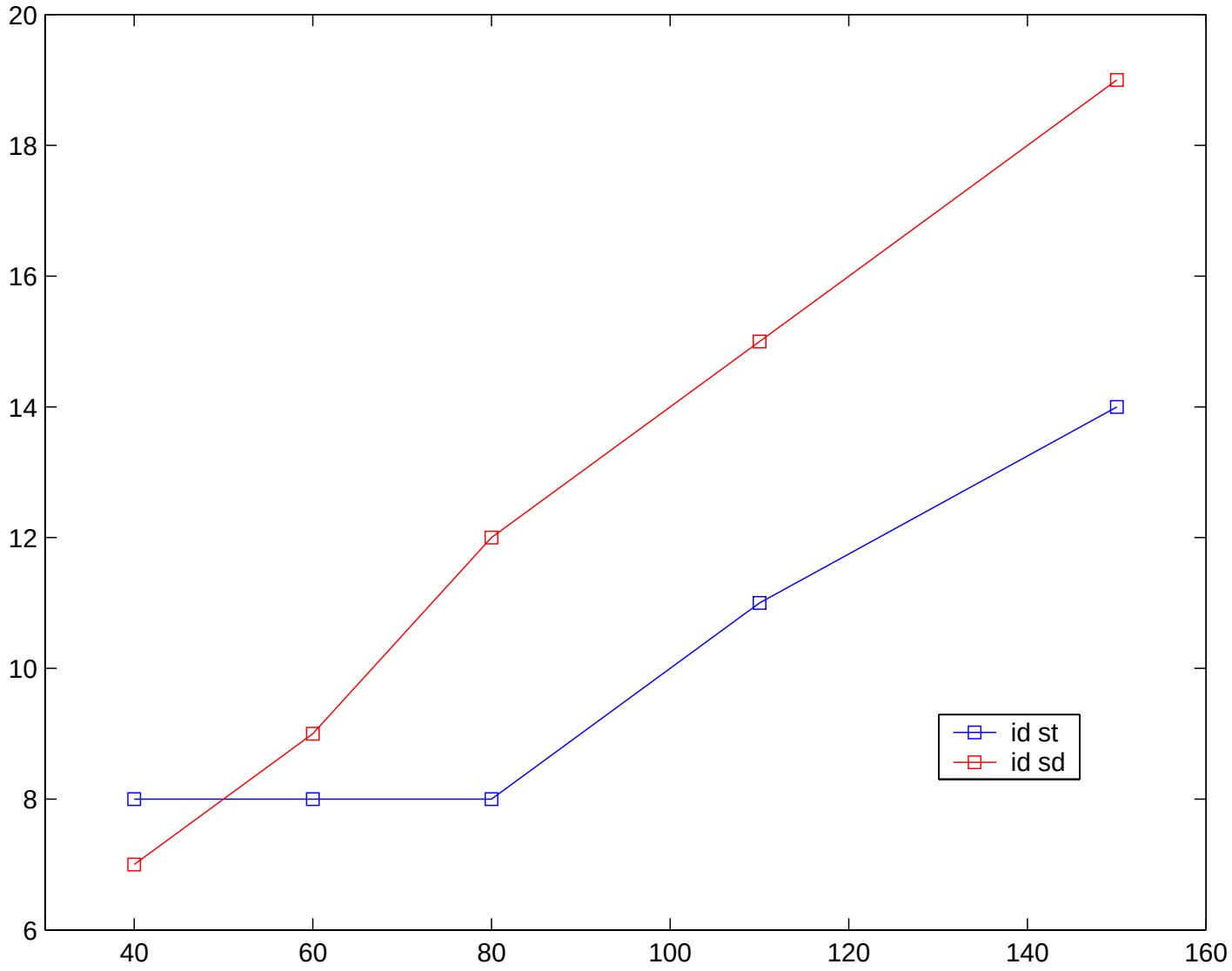
Scalability for the Poisson problem, number of iterations



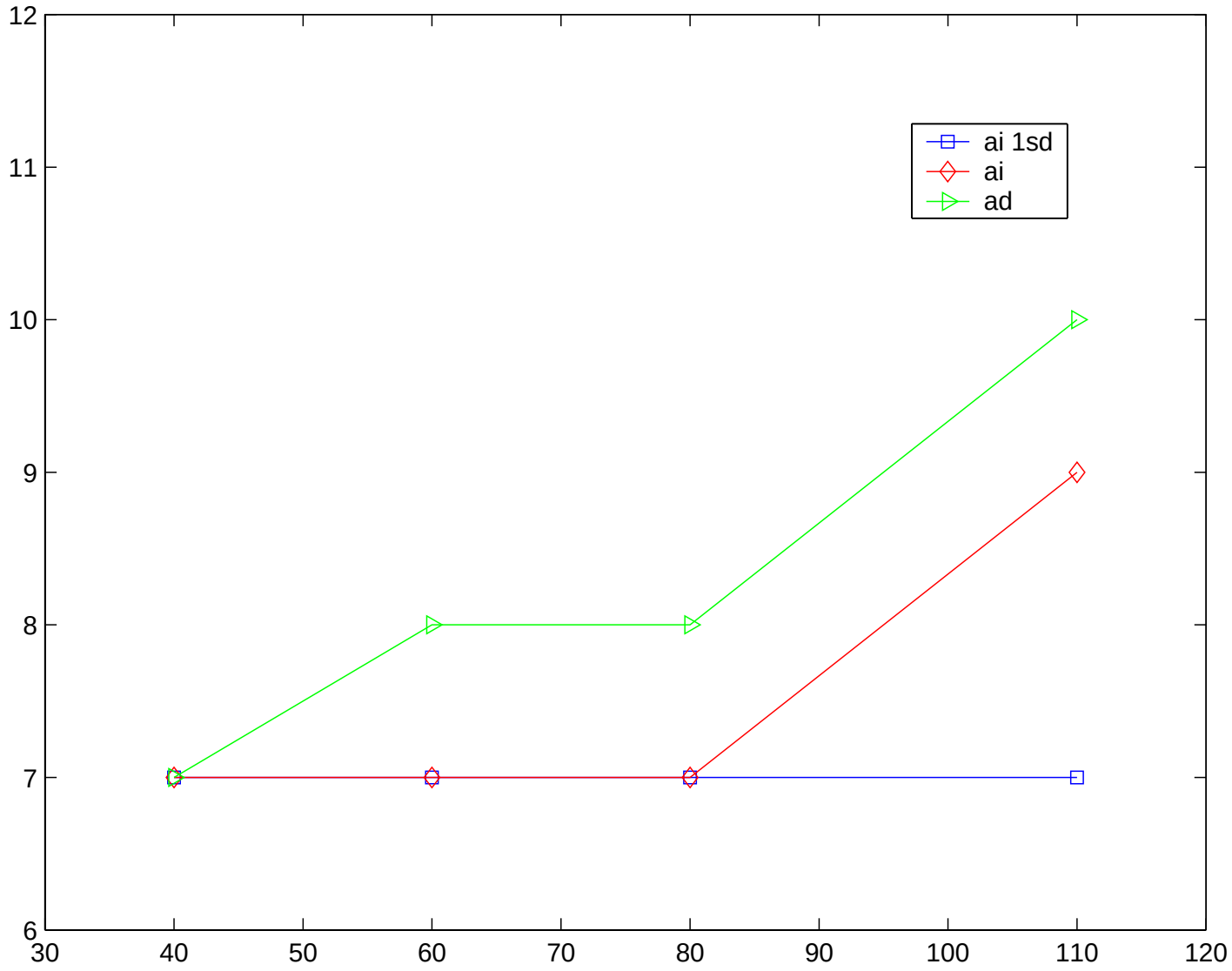
Scalability for the Poisson problem, number of iterations when keeping some fill in



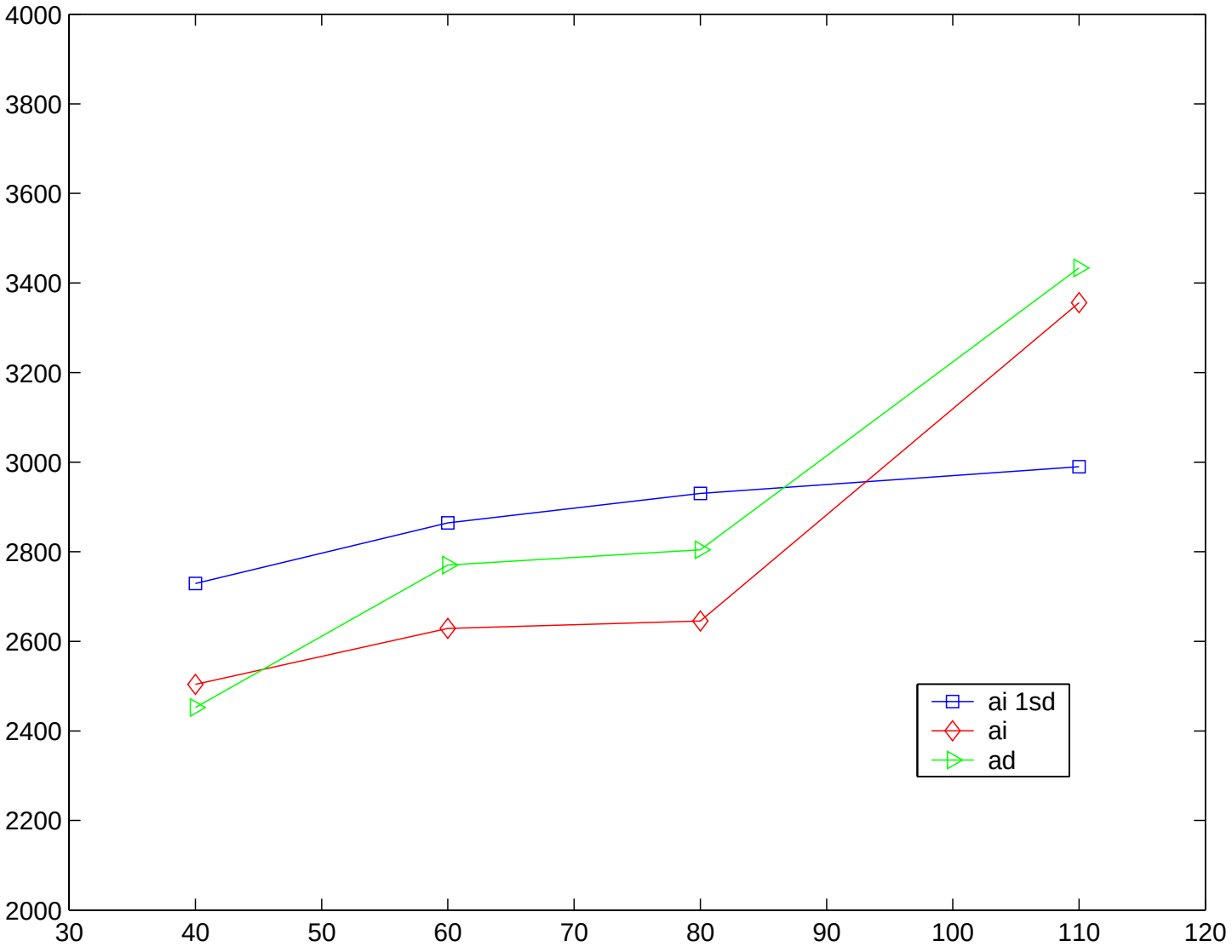
Scalability for the Poisson problem, number of operations divided by n



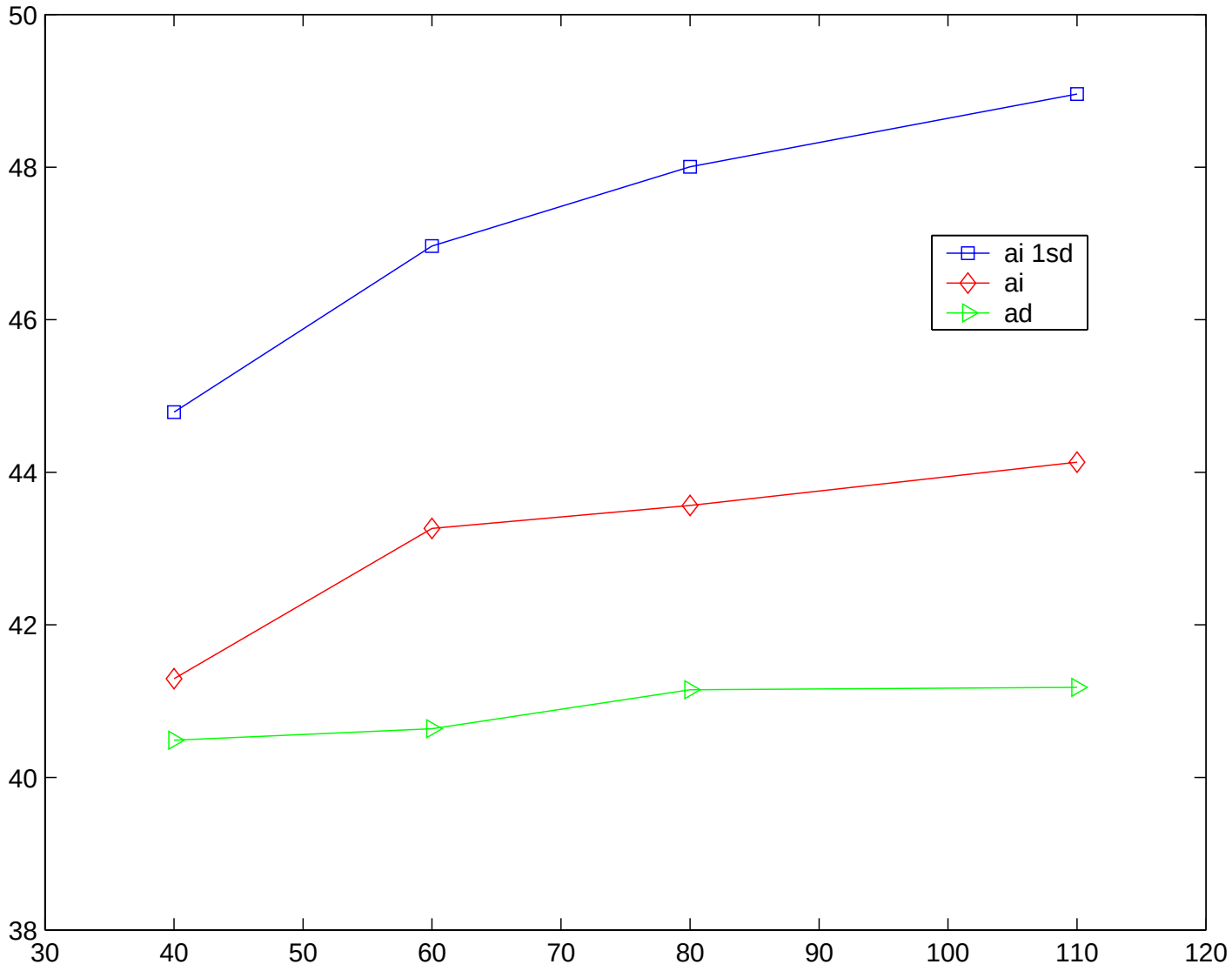
Scalability for the Poisson problem, number of iterations with different coarsenings



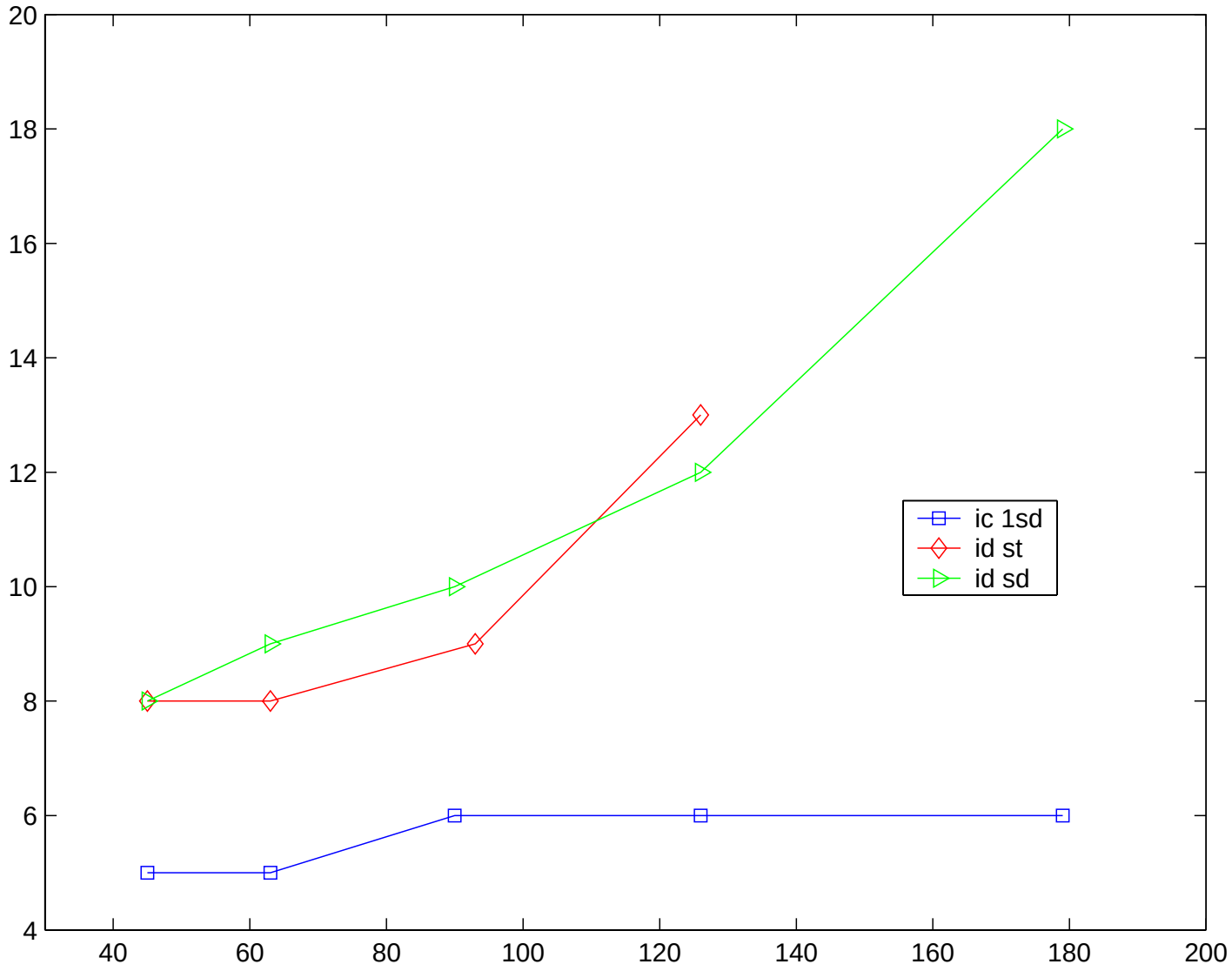
Scalability for the Poisson problem, number of iterations



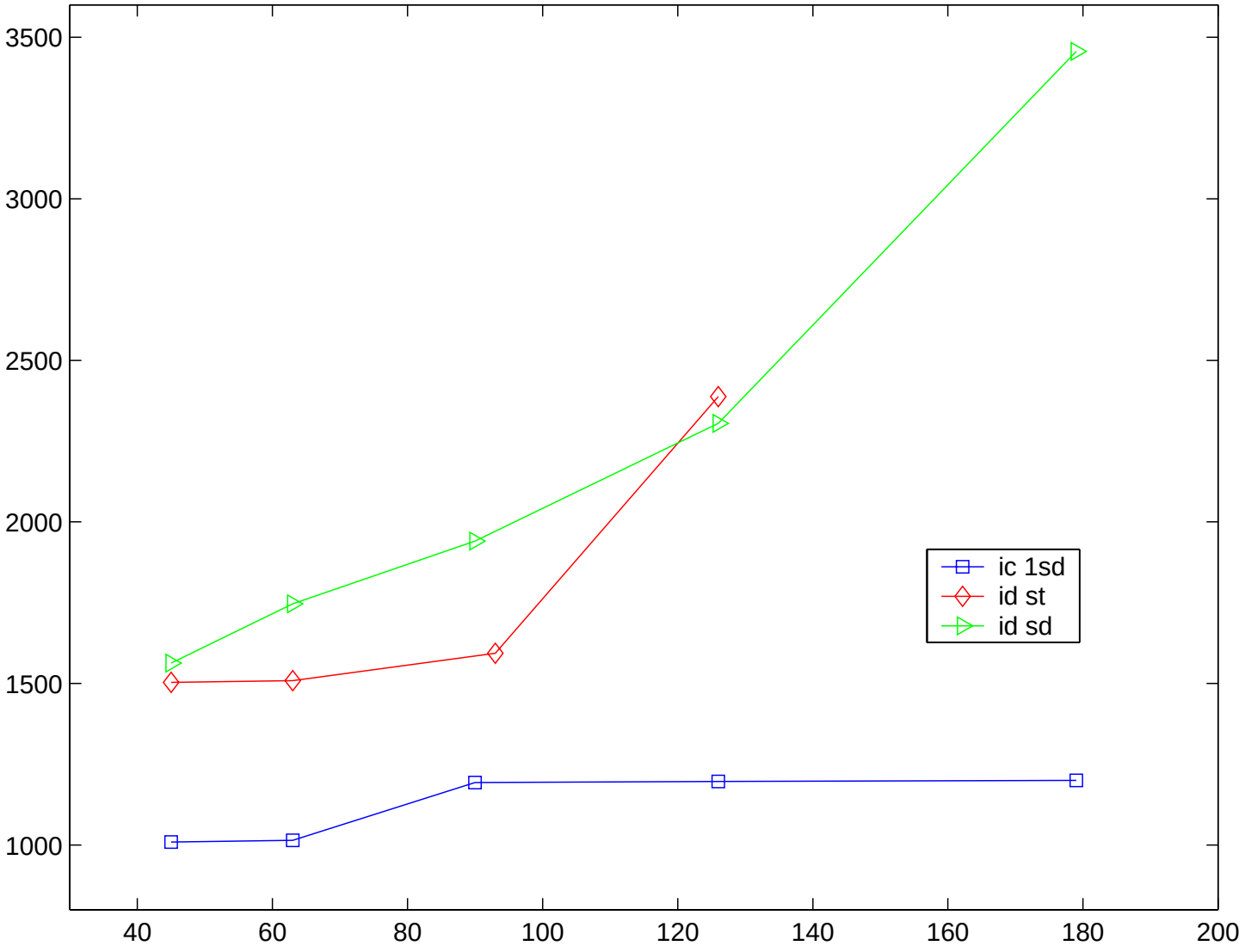
Scalability for the Poisson problem, number of operations divided by n



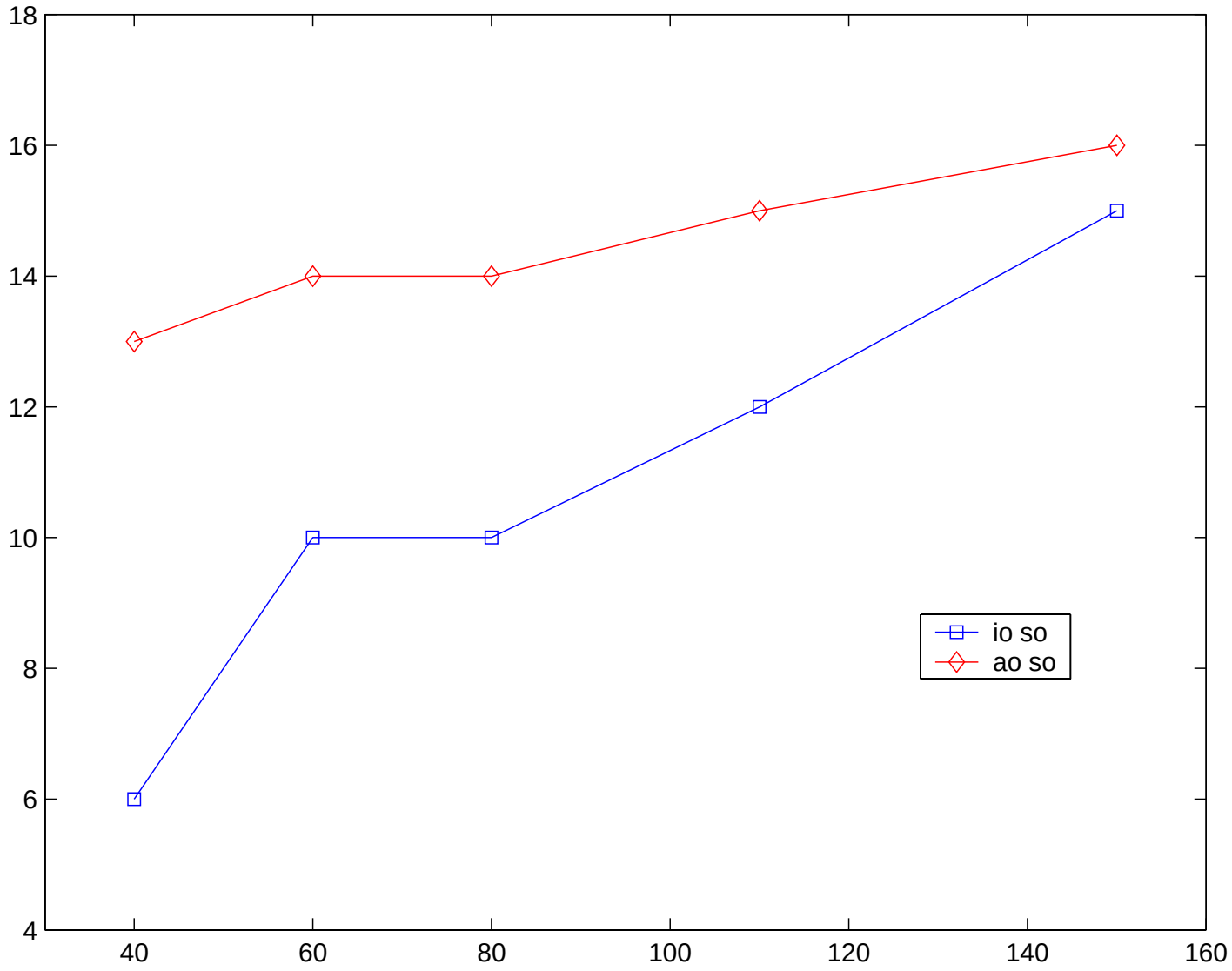
Scalability for the Poisson problem, storage divided by n



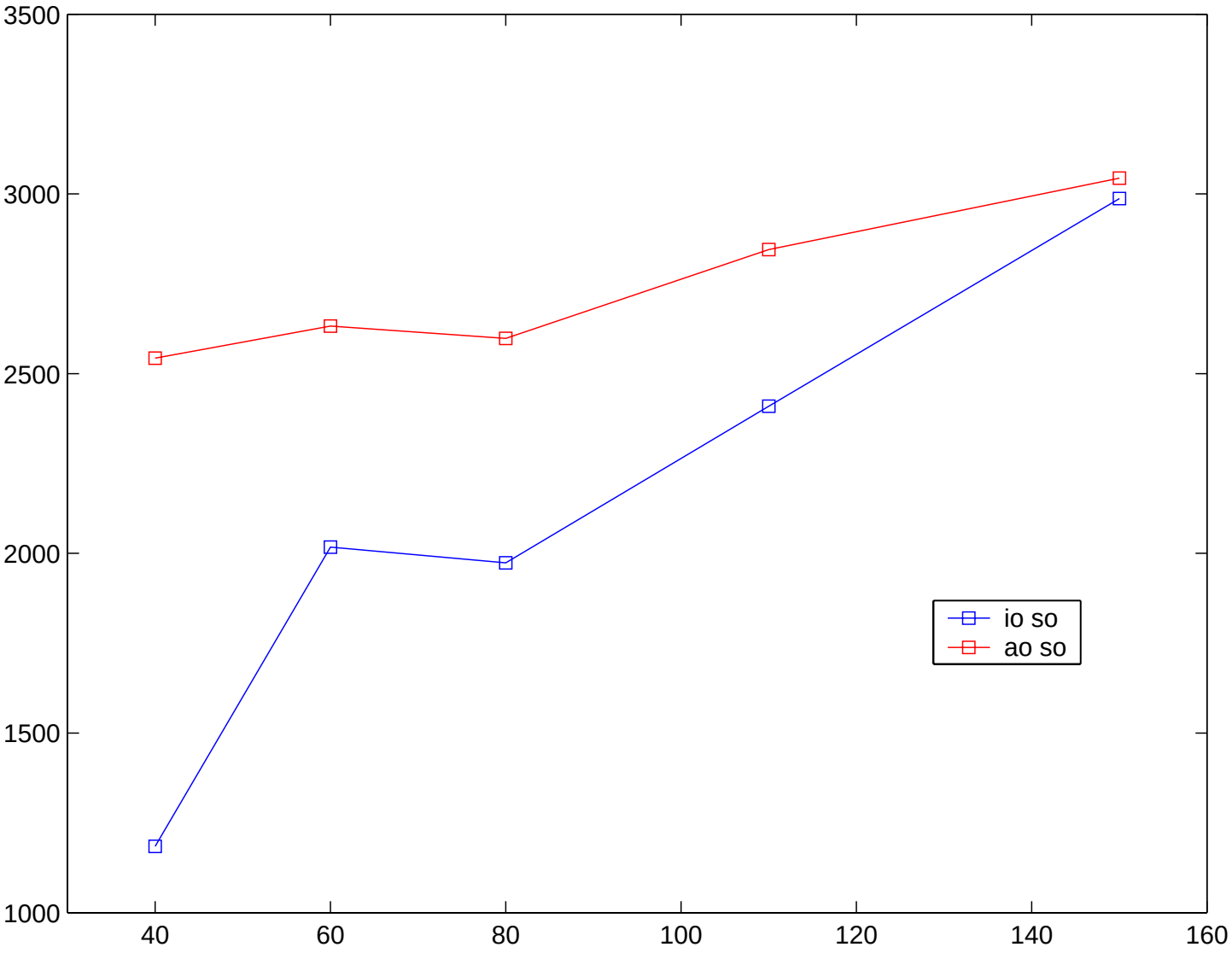
Scalability for the Poisson problem, number of iterations



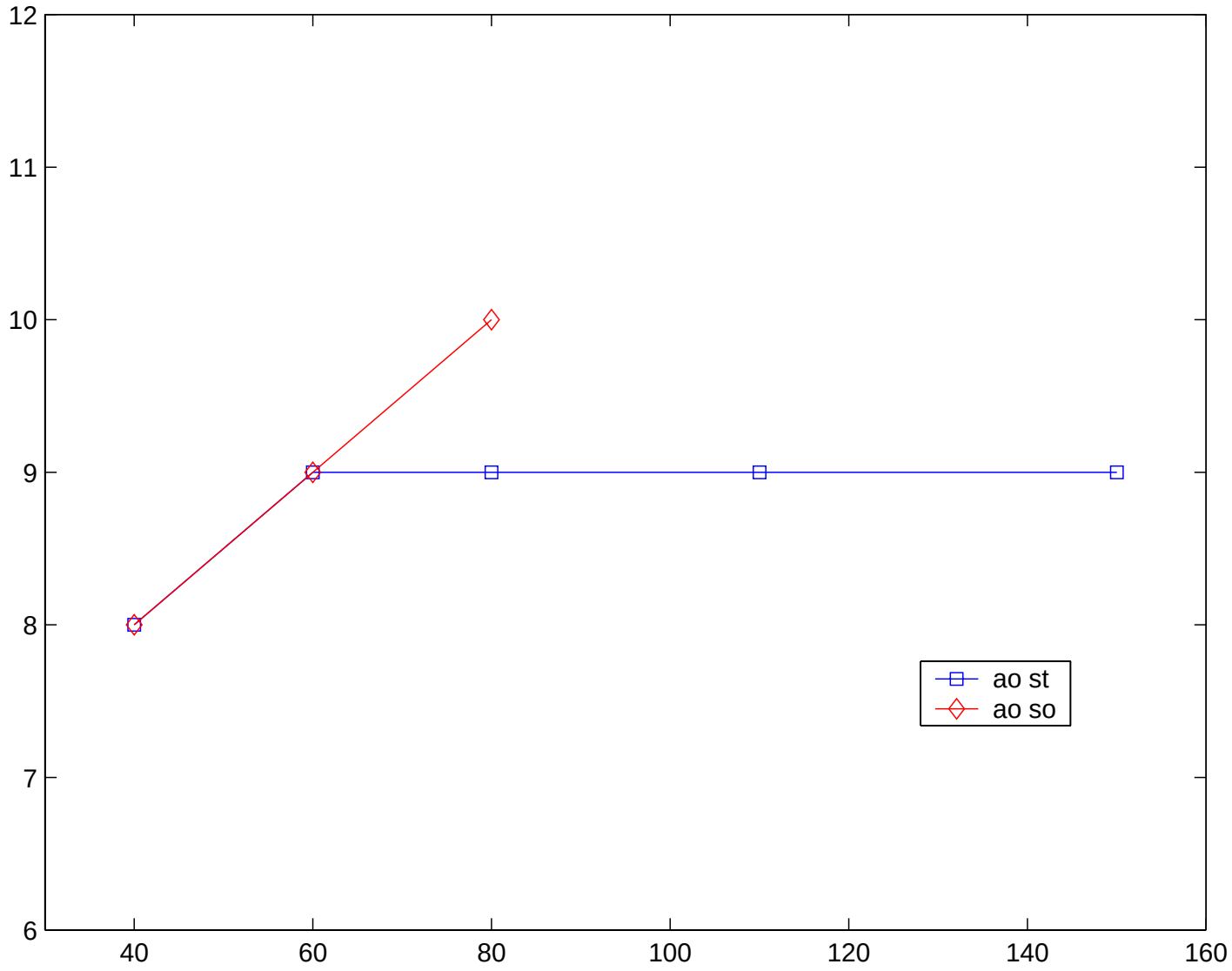
Scalability for the Poisson problem, number of operations divided by n



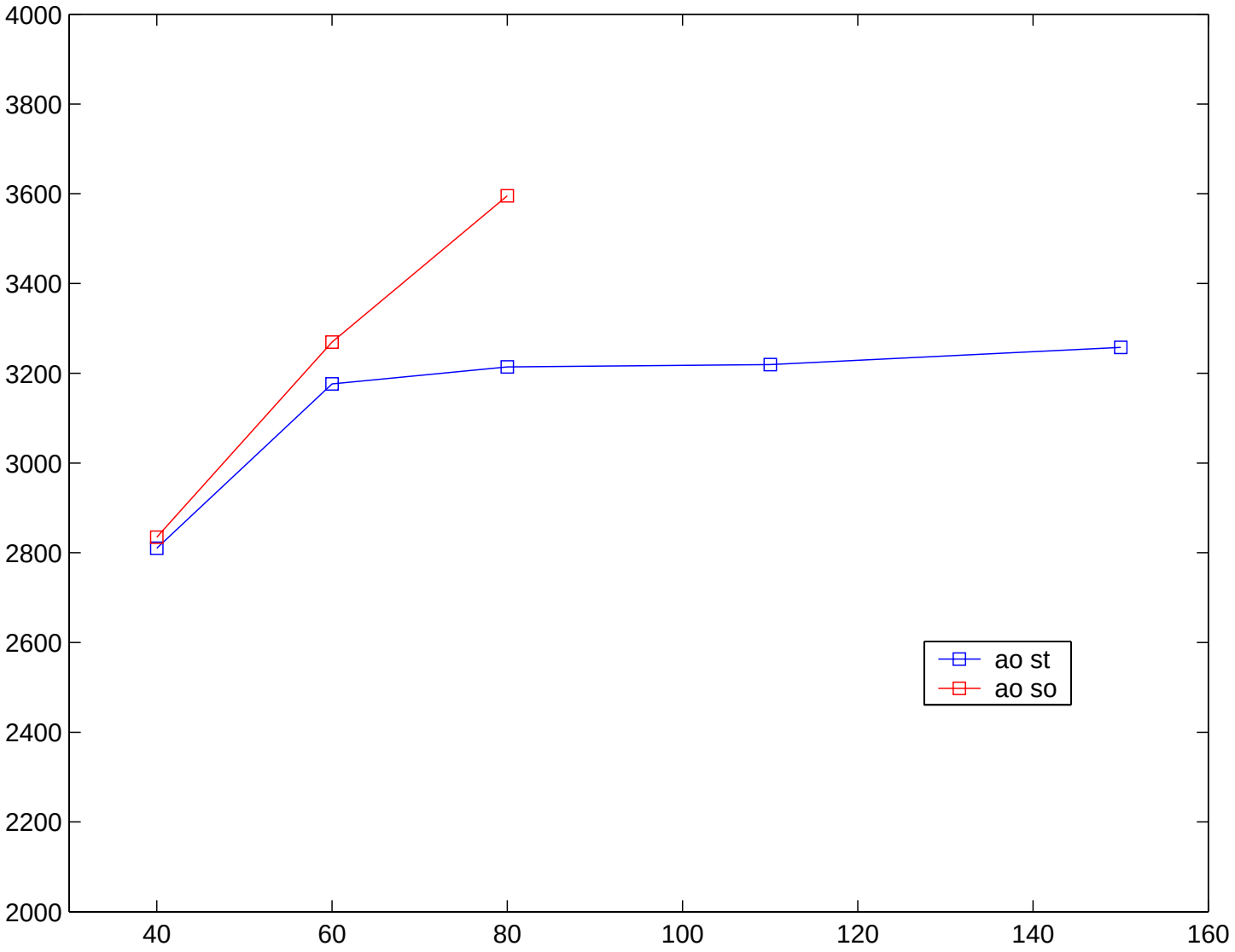
Scalability for the Poisson problem, number of iterations



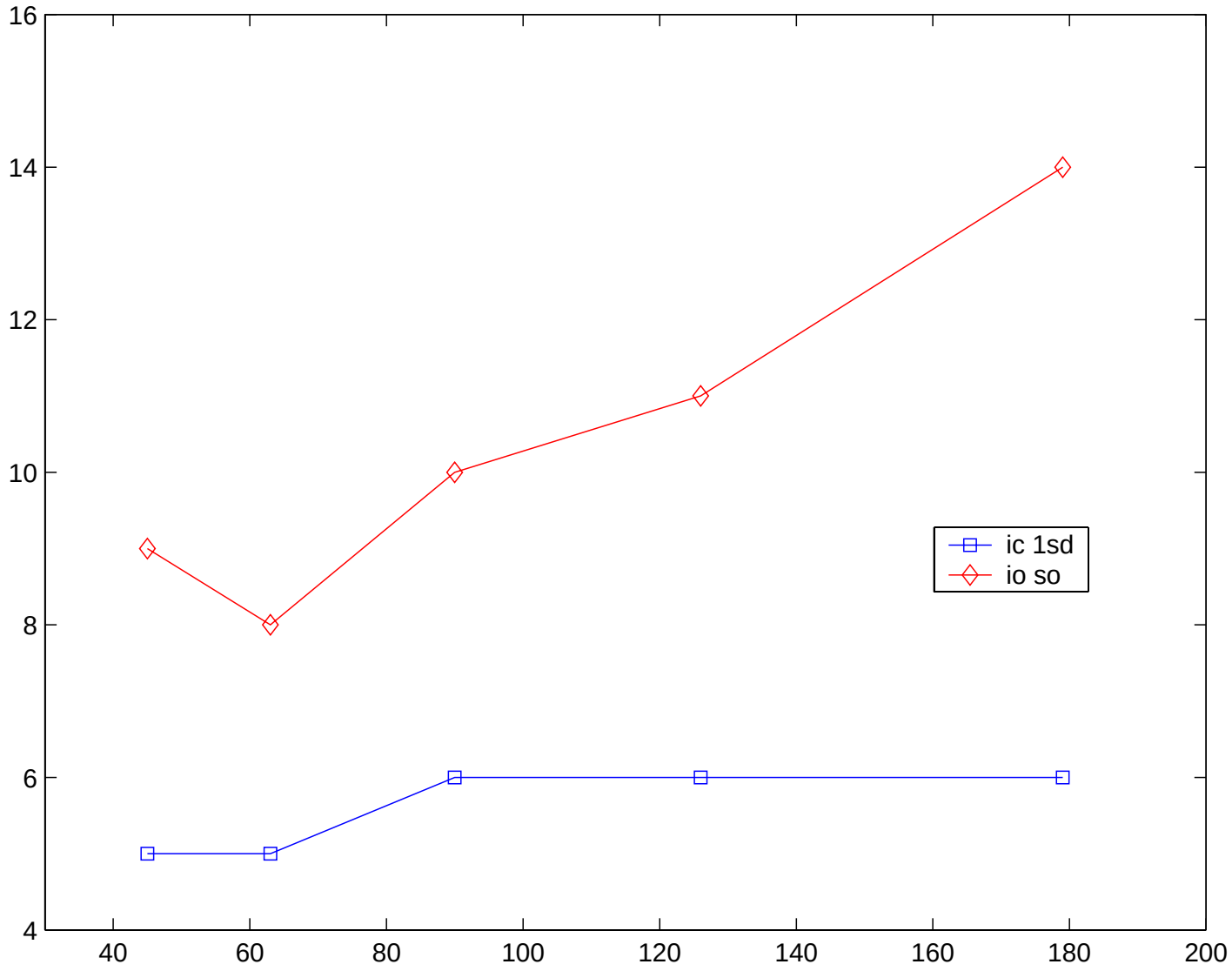
Scalability for the Poisson problem, number of operations/n



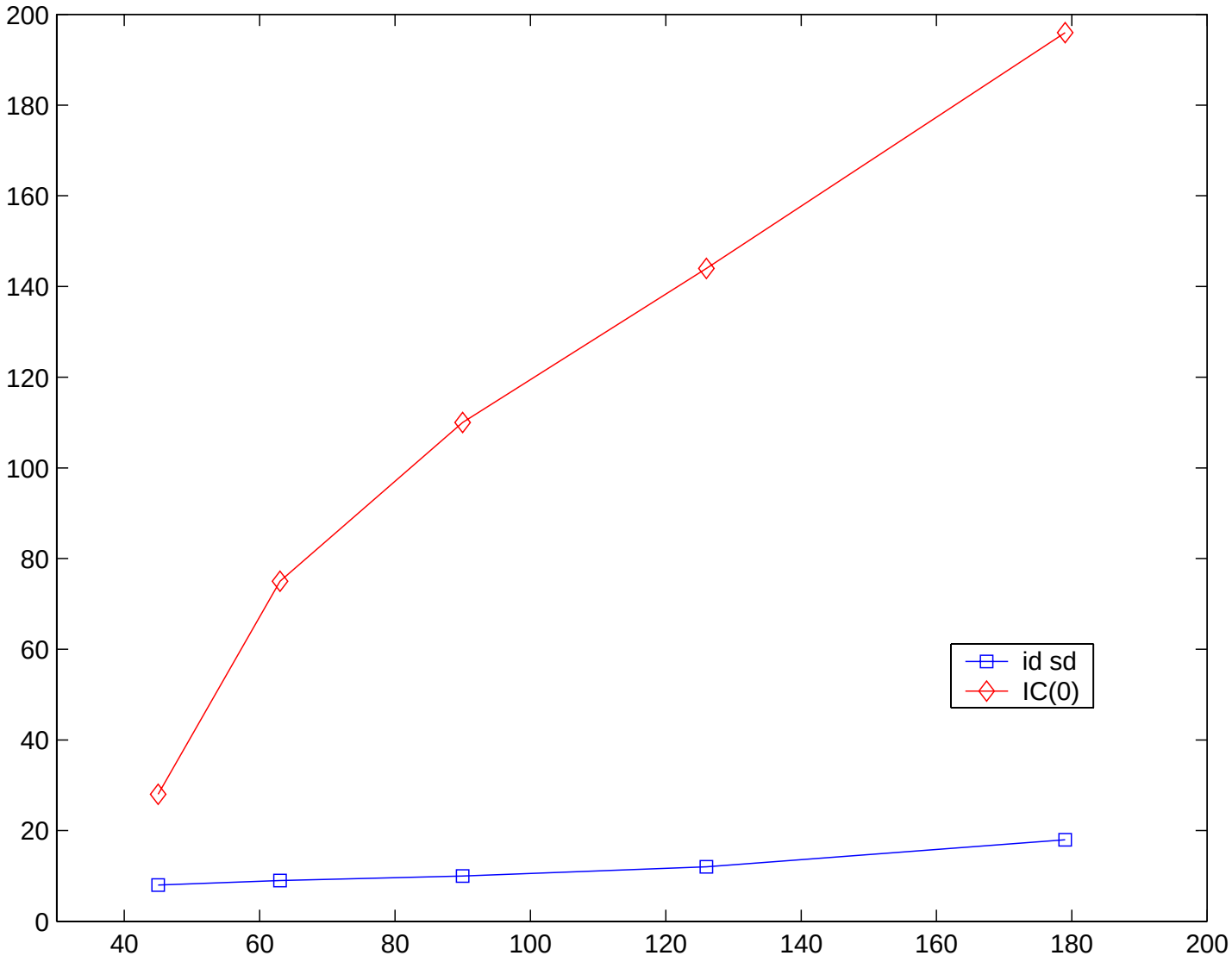
Scalability for the Poisson problem, number of iterations, normalized matrix



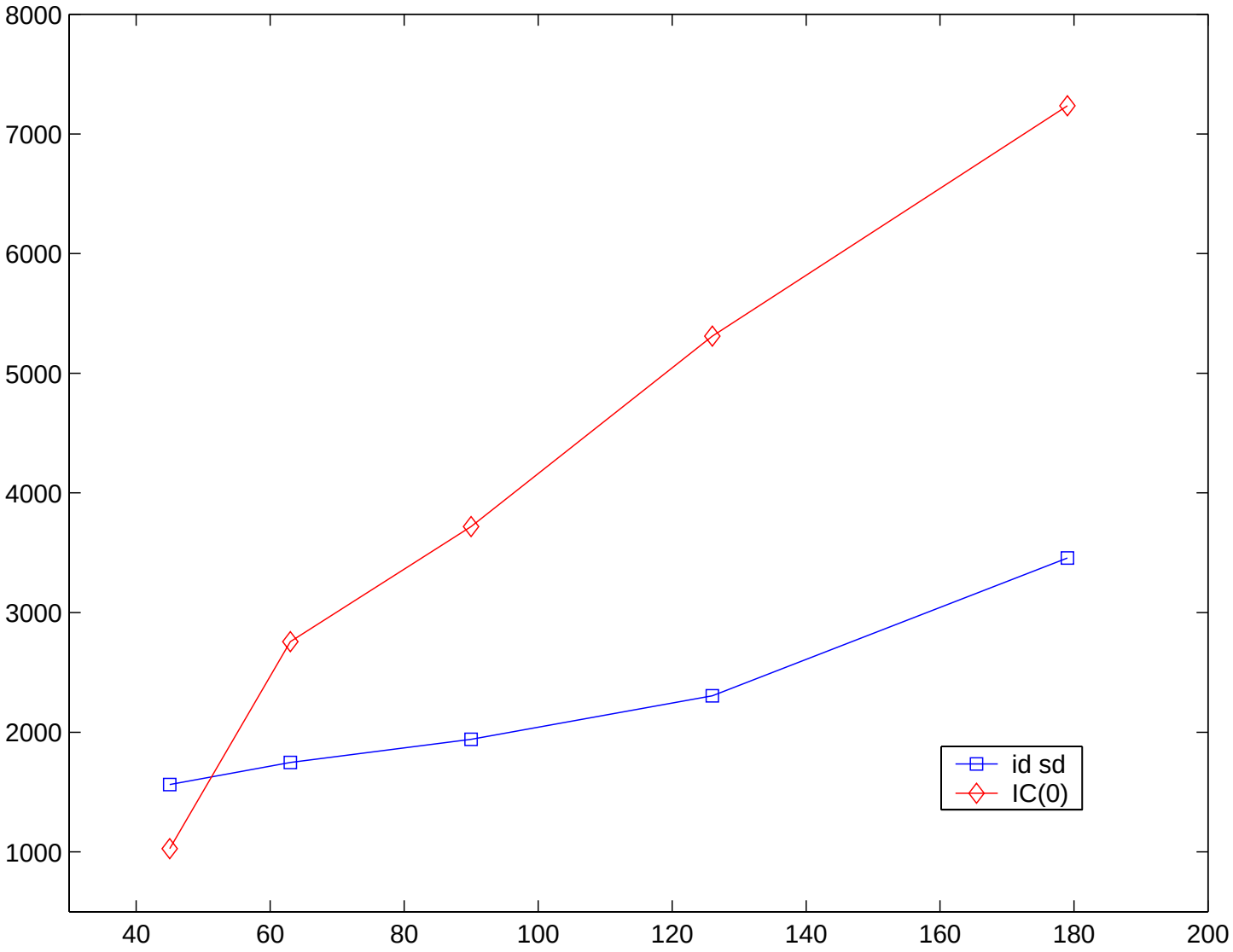
Scalability for the Poisson problem, number of operations/n, normalized matrix



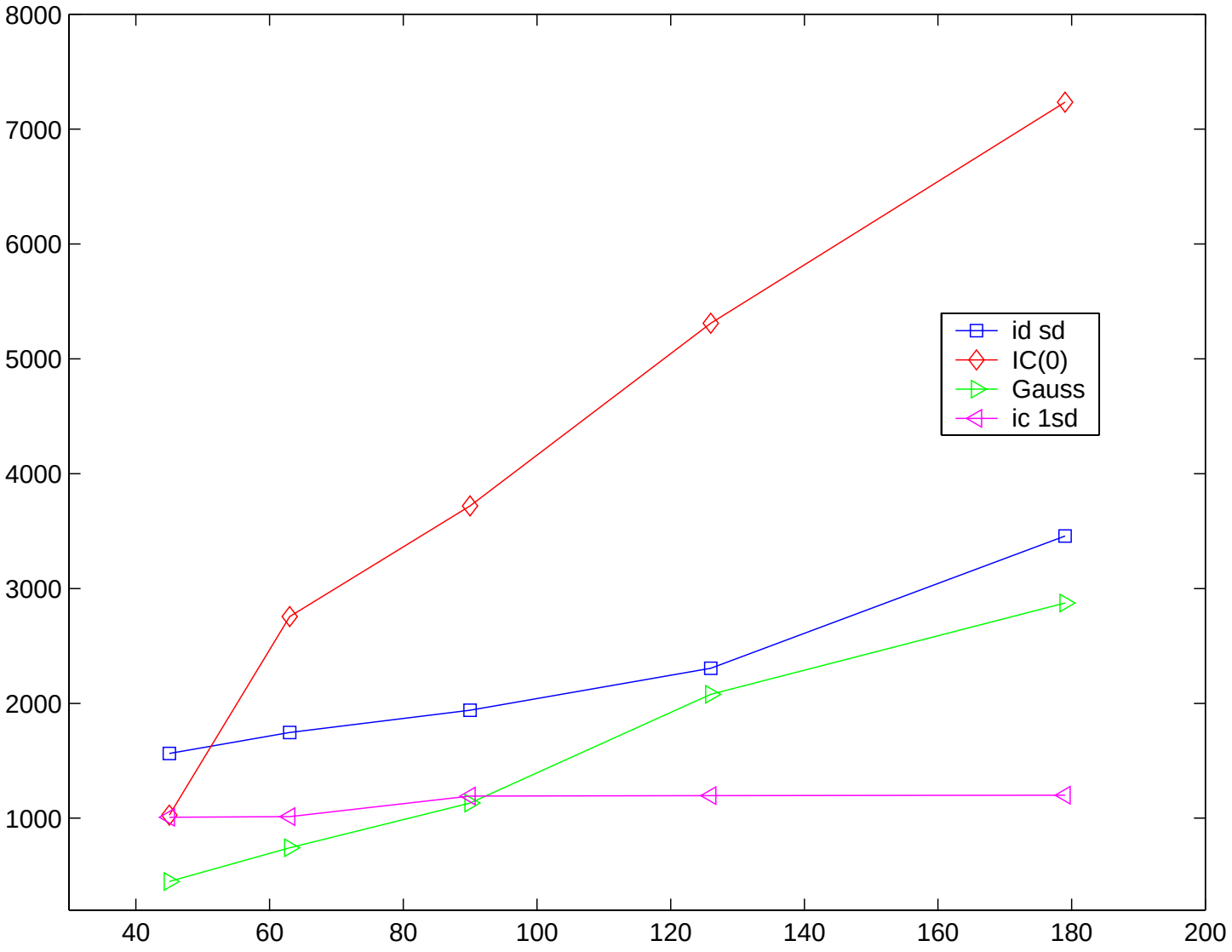
Scalability for the Poisson problem, number of iterations



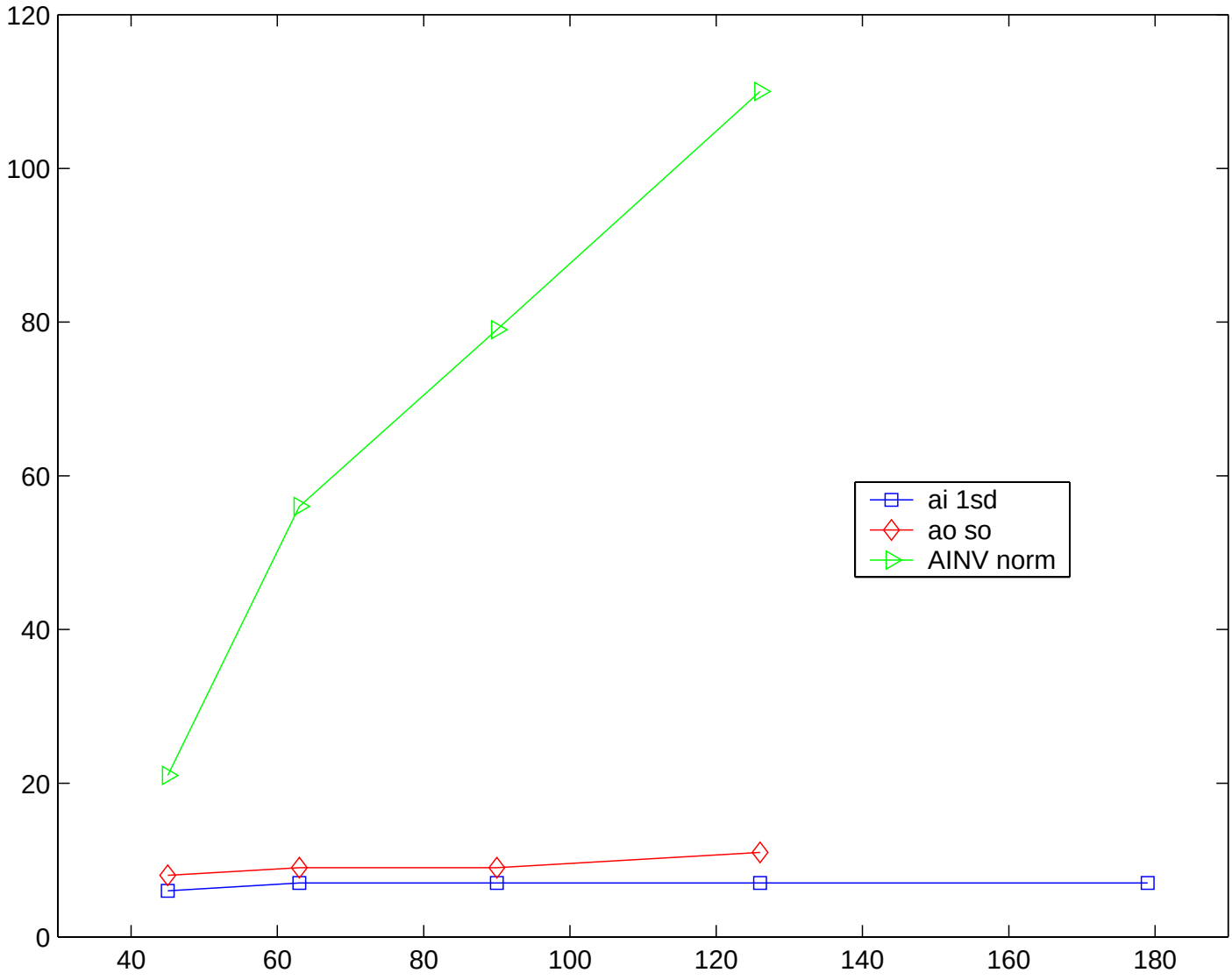
Scalability for the Poisson problem, number of iterations



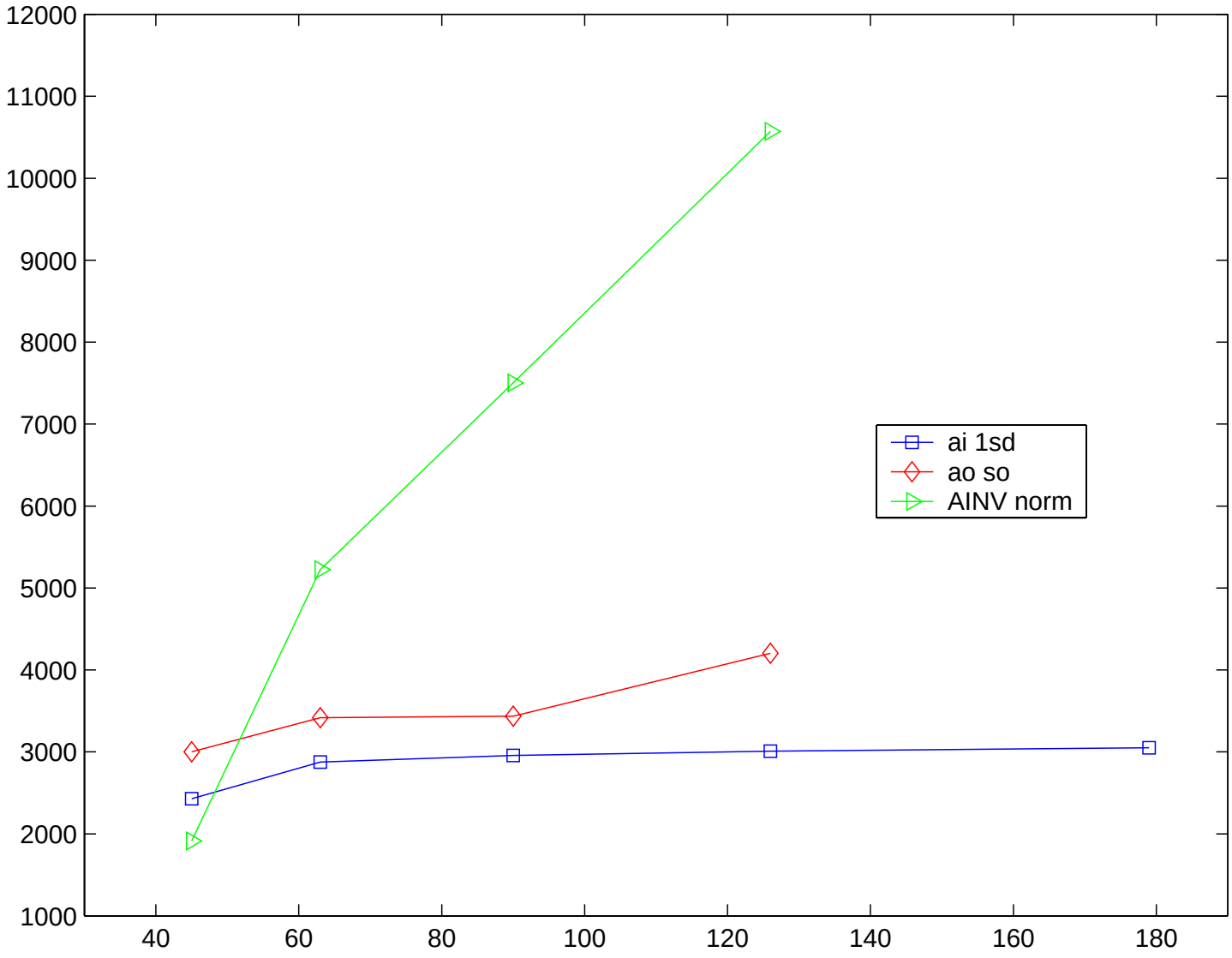
Scalability for the Poisson problem, number of operations/n



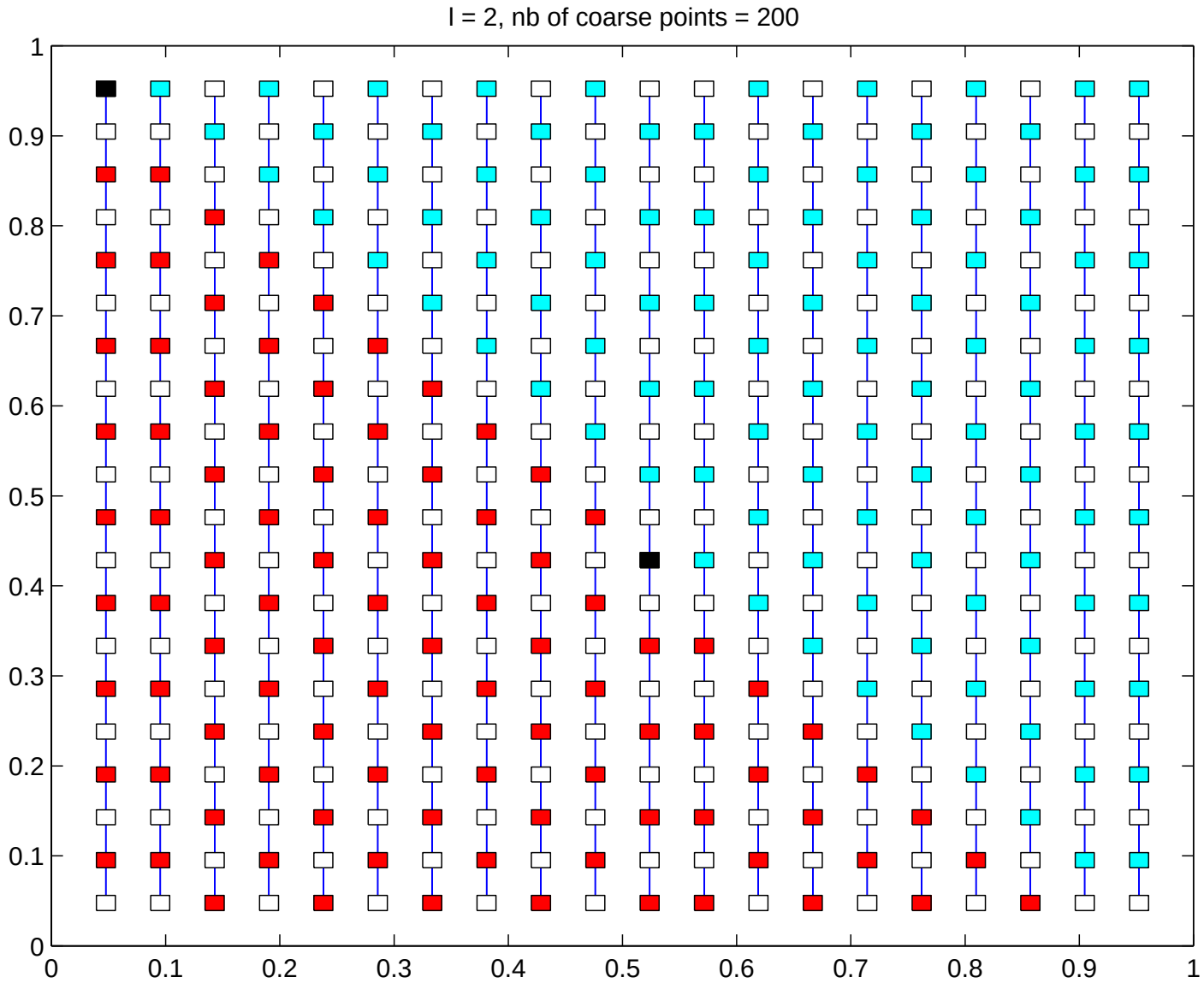
Scalability for the Poisson problem, number of operations/n



Scalability for the Poisson problem, number of iterations



Scalability for the Poisson problem, number of operations/n



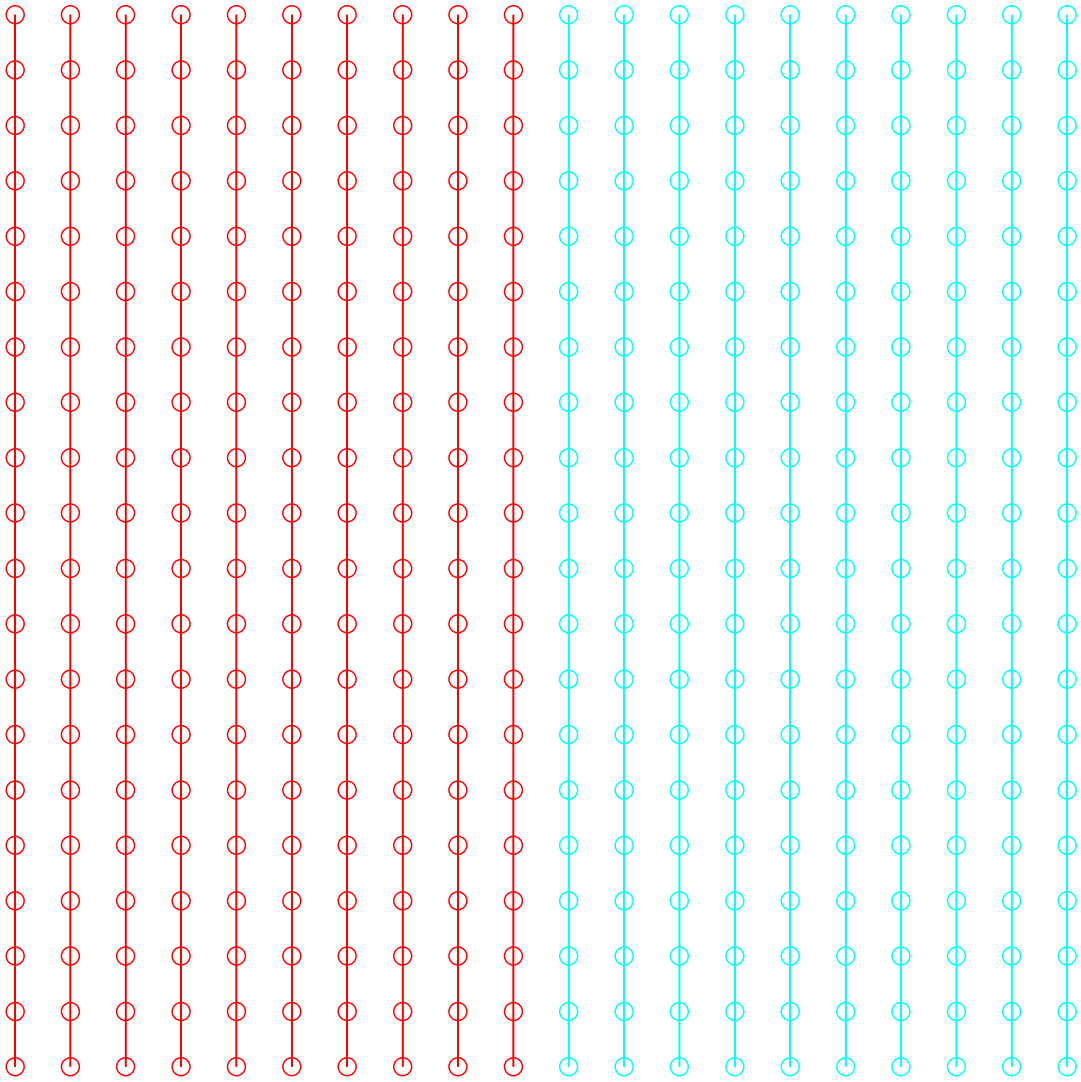
RCM partitioning of the anisotropic problem matrix

level 2

Problem

We throw away some fill–in between nodes which are physically strongly connected

Therefore, it is better to partition the influence matrix S

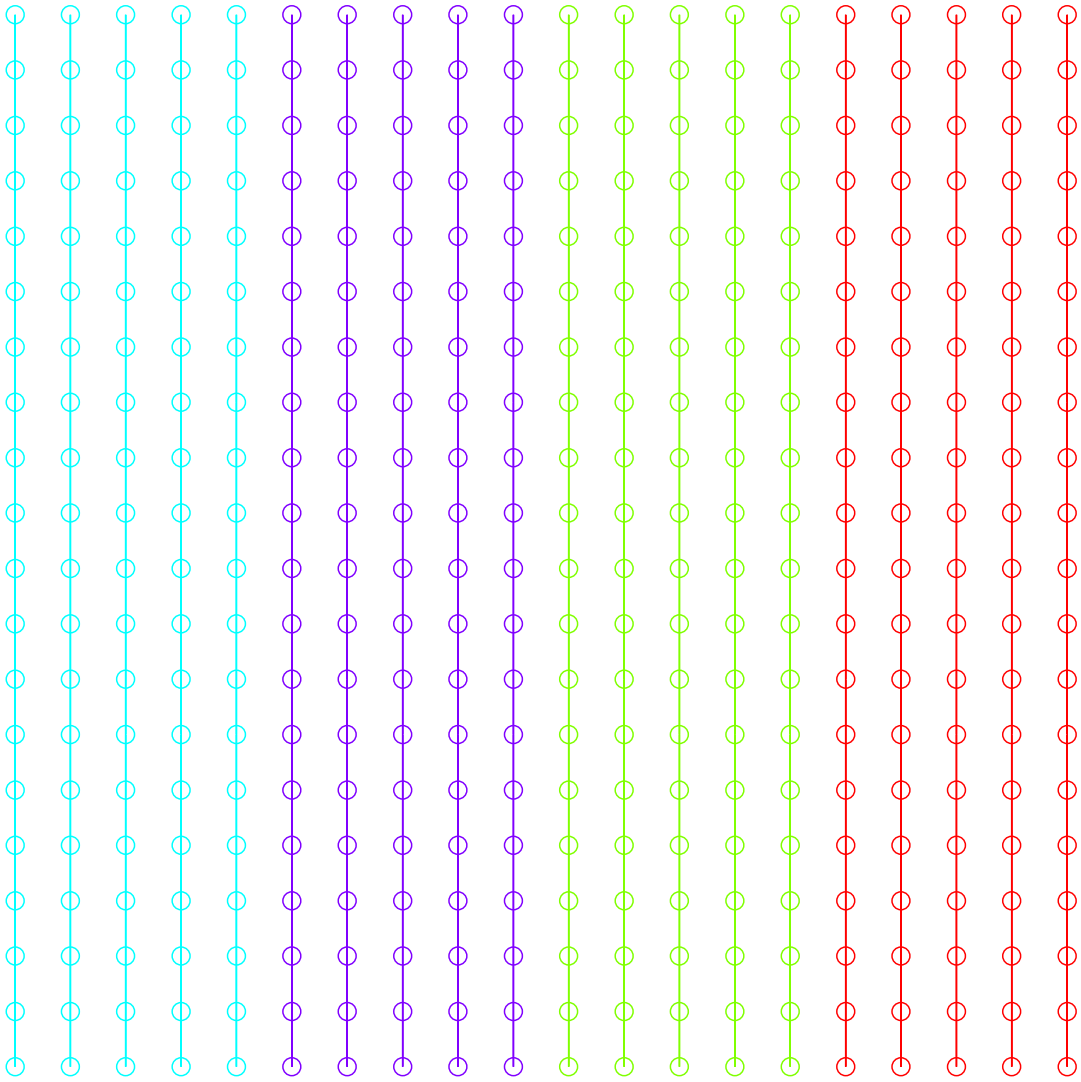


0 cut edges

RCM partitioning of S for the anisotropic problem matrix

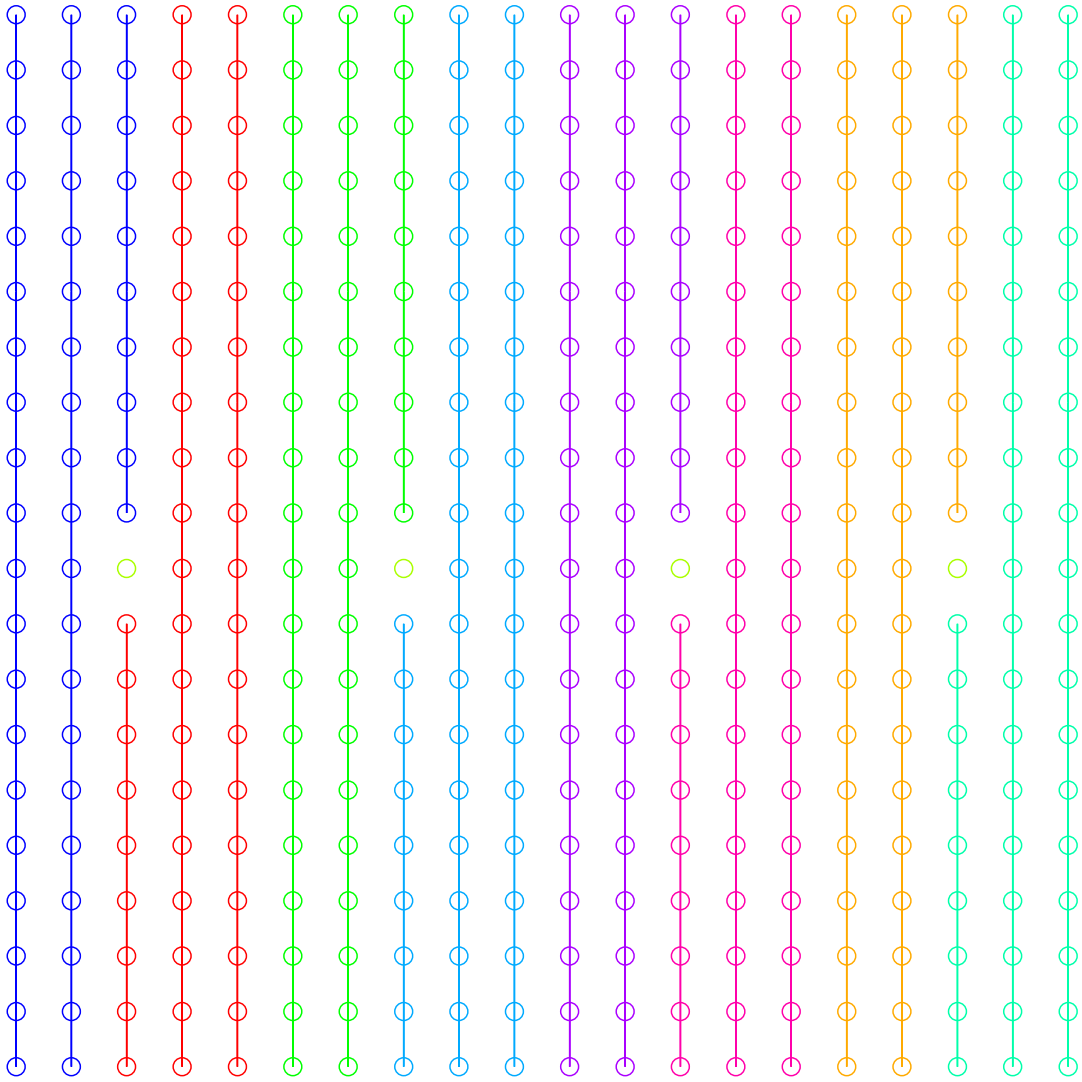
RCM partitioning of the anisotropic problem matrix

level 2



0 cut edges

RCM partitioning of the anisotropic problem matrix



8 cut edges

RCM partitioning of the anisotropic problem matrix

PCG for the anisotropic problem, $m = 40$, $\tau = 0.05$

multilevel, ('id', 'b', 'st', 'st')

nb sd	nb it	flops	storage
		Part of A	
1	4	1 555 936	41 417
2	11	3 466 893	37 630
4	13	3 990 023	37 066
8	16	4 836 593	36 992
16	17	5 105 097	36 915
32	19	5 696 773	37 093
		Part of S	
2	4	1 556 065	41 419
4	4	1 541 082	41 046
8	4	1 504 433	40 129
16	5	1 748 607	38 832
32	7	2 168 453	36 247

PCG for the anisotropic problem, $m = 40$, $\tau = 0.05$

multilevel, ('id', 'b', 'sd', 'st'), Part of S

nb sd	nb it	flops	storage
2	4	1 546 296	41 152
4	4	1 532 286	40 808
8	4	1 482 198	39 574
16	5	1 739 221	38 609
32	5	1 615 497	36 105

PCG for the anisotropic problem, $m = 60$, $\tau = 0.05$

multilevel, ('id', 'b', 'sd', 'st'), Part of S

nb sd	nb it	flops	storage
2	4	3 571 098	95 322
4	4	3 530 448	94 214
8	5	4 174 573	92 628
16	5	4 014 895	89 044
32	5	3 695 124	81 085

PCG for the anisotropic problem, $m = 40$, $\tau = 0.05$

multilevel, ('ad', 'b', 'sd', 'st'), Part of S

nb sd	nb it	flops	storage
2	5	3 649 833	79 984
4	5	3 590 301	78 750
8	5	3 447 363	75 769
16	5	3 152 583	69 606
32	6	3 208 571	61 160

PCG for the anisotropic problem, $m = 60$, $\tau = 0.05$

multilevel, ('ad', 'b', 'sd', 'st'), Part of S

nb sd	nb it	flops	storage
2	5	9 744 979	212 324
4	5	9 563 935	208 453
8	5	9 144 157	199 657
16	6	9 847 154	184 505
32	6	8 734 777	163 256

PCG for the anisotropic problem, $m = 40$, $\tau = 0.05$

multilevel, ('io', 'b', 'so', 'st'), Part of S

nb sd	nb it	flops	storage
2	4	1 546 655	41 152
4	4	1 532 547	40 808
8	4	1 486 198	39 574
16	9	2 931 363	38 628
32	10	3 086 887	36 547

PCG for the anisotropic problem, $m = 60$, $\tau = 0.05$

multilevel, ('io', 'b', 'so', 'st'), Part of S

nb sd	nb it	flops	storage
2	4	3 570 958	95 322
4	4	3 530 268	94 214
8	8	6 279 145	92 274
16	9	6 710 701	88 355
32	9	6 462 752	82 173

Matrices from the Matrix Market

- 1138-bus admittance matrix
- nos 7

use scaling

PCG 1138-bus, $n = 1138$, $\tau = 0.05$

multilevel, ('id', 'b', 'sd', 'st')

nb sd	nb it	flops	storage
1	38	6 452 066	21 105
2	37	6 810 001	23 244
4	38	6 407 725	20 965
8	37	6 596 076	22 370
16	36	5 831 250	19 498
32	35	6 193 153	21 033
IC	163	5 802 060	3734
AINV 0.05	85	5 190 500	10667
SAINV 0.05	74	7 572 900	11585

PCG 1138-bus, $n = 1138$, $\tau = 0.05$

multilevel, ('po', 'b', 'so', 'st')

nb sd	nb it	flops	storage
1	113	25 316 337	13 127
2	126	29 537 098	13 832
4	116	29 379 519	15 186
8	106	27 613 197	15 664
16	103	28 445 105	16 698
32	103	29 917 537	17 697
IC	163	5 802 060	3734
AINV 0.05	85	5 190 500	10667
SAINV 0.05	74	7 572 900	11585

A must be shifted

PCG 1138-bus, $n = 1138$, $\tau = 0.05$

multilevel, ('ad', 'b', 'sd', 'st')

nb sd	nb it	flops	storage
1	51	15 590 289	38 859
2	52	14 361 966	35 715
4	71	18 135 801	32 875
8	62	16 193 772	33 802
16	82	19 780 708	30 488
32	56	14 147 586	31 484
IC	163	5 802 060	3734
AINV 0.05	85	5 190 500	10667
SAINV 0.05	74	7 572 900	11585

PCG 1138-bus, $n = 1138$, $\tau = 0.05$

multilevel, ('ao($A + 0.1I$)', 'b', 'so', 'st')

nb sd	nb it	flops	storage
1	63	11 249 873	23 347
2	67	15 203 853	29 514
4	66	17 680 851	34 988
8	65	17 635 689	35 527
16	65	17 874 345	36 233
32	62	17 622 927	37 576
IC	163	5 802 060	3734
AINV 0.05	85	5 190 500	10667
SAINV 0.05	74	7 572 900	11585

PCG Nos7, $n = 729$, $\tau = 0.05$

multilevel, ('id', 'b', 'sd', 'st')

nb sd	nb it	flops	storage
1	9	1 840 612	25 118
2	9	1 711 554	23 255
4	12	2 536 339	26 812
8	11	2 397 621	27 807
16	19	3 753 364	26 172
32	18	4 231 524	14 961

PCG Nos7, $n = 729$, $\tau = 0.05$

multilevel, ('ad', 'b', 'sd', 'st')

nb sd	nb it	flops	storage
1	21	4 300 819	27 059
2	19	3 672 067	25 331
4	19	4 043 607	28 312
8	17	3 680 959	29 001
16	14	2 874 982	27 325
32	18	4 530 488	17 287

PCG Nos7, $n = 729$, $\tau = 0.05$

multilevel, ('ao', 'b', 'so', 'st')

nb sd	nb it	flops	storage
1	21	4 300 819	27 059
2	22	6 226 664	36 733
4	20	7 106 551	45 623
8	19	6 785 267	45 953
16	19	6 918 907	47 041
32	17	6 074 419	46 026

PCG Nos7, $n = 729$, $\tau = 0.05$

multilevel, ('po', 'b', 'so', 'st')

nb sd	nb it	flops	storage
2	14	3 978 832	17 472
4	19	5 748 867	18 998
8	17	5 319 139	19 746
16	17	5 559 799	20 708
32	17	5 570 275	20 770

Possible improvements

- For IC use the algorithms from
    - Magolu Monga Made and Van der Vorst
    - Pothen and Hysom

However they both use a dd like ordering of the unknowns

Find a good fully parallel smoother ?

Conclusion

We have an algorithm which is simple and relatively easy to code, the number of iterations is not growing too fast on most problems

Sometimes better to partition the graph of the influence matrix

There is parallelism in the construction and solution phases for the smoothers

For parallelism AINV seems better than IC

There are still problems to solve on parallel computers:

- load balancing on coarse levels?
- how many levels to use?
- enough work to do on coarse levels?