DDαAMG for the Schwinger model

We document and explain in detail the implementation of the DDαAMG solver for the 2-flavor Schwinger model. The code is available on GitHub.

1 Schwinger model

The Schwinger model represents Quantum Electrodynamics (QED) in two dimensions: one space and one time dimension. The model shares similar properties with QCD, such as confinement, chiral symmetry breaking and their topology. Mathematically, the Schwinger model has less degrees. For these reasons, it is a good *toy model* for testing numerical methods which could be applied to QCD as well. We define the Dirac matrix for the Schwinger model in this section.

Let us define the $2d$ lattice as

$$\Lambda := \{n = (n_t, n_x) \,|\, n_t = 0, 1, \dots, N_t - 1;\; n_x = 0, 1 \dots, N_x - 1\}. \tag{1}$$

At each site there are two *spin* components, *i.e.* at each site we consider two complex numbers. Therefore, a vector defined on the lattice has the form $\psi_\mu(n)$, where $\mu = 0, 1$ and $n \in \Lambda$. Then, we have a total of $2N_xN_t$ complex variables. Thus, the set of variables is

$$\mathcal{V} = \Lambda \times \{0, 1\}. \tag{2}$$

The Dirac-Wilson operator (in lattice units) is defined as

$$\begin{aligned} D_{\alpha\beta}[\,n, n'\,] &= (m_0 + 2)\delta^{\alpha\beta}\delta_{n,n'} \\ &\quad - \frac{1}{2}\sum_{\mu=0}^{1}\left[(1 - \sigma_\mu)^{\alpha\beta}\, U_\mu(n)\delta_{n+\hat{\mu},n'} + (1 + \sigma_\mu)^{\alpha\beta}\, U_\mu^\dagger(n - \hat{\mu})\delta_{n-\hat{\mu},n'}\right]. \end{aligned} \tag{3}$$

The index $\mu = 0$ refers to time and $\mu = 1$ to space and m_0 is the *bare mass.* The vector $\hat{\mu}$ represents a unit vector in the direction indexed by μ. We choose the following representation for σ

$$\sigma_0 = \begin{pmatrix} 0 & 1 \\ 1 & 0 \end{pmatrix}, \quad \sigma_1 = \begin{pmatrix} 0 & -i \\ i & 0 \end{pmatrix}. \tag{4}$$

The *link variables* $U_\mu(n) \in \mathrm{U}(1)$ are randomly generated during simulations. We consider periodic boundaries for the space direction and antiperiodic for the time direction, *i.e.* for any field $\psi_\mu(n)$ on the lattice we have

$$\psi_\mu\left[(N_x - 1 + 1, n_t)\right] = \psi_\mu\left[(0, n_t)\right], \quad \psi_\mu\left[(n_x, N_t - 1 + 1)\right] = -\psi_\mu\left[(n_x, 0)\right]. \tag{5}$$

The application of D to a field on the lattice is

$$\begin{aligned}(D\psi)_\alpha(n) &= (m_0+2)\psi_\alpha(n) \\ &\quad - \frac{1}{2}\sum_{\mu,\beta=0}^{1}\Big[(1-\sigma_\mu)^{\alpha\beta}\,U_\mu(n)\,\psi_\beta(n+\hat{\mu}) + (1+\sigma_\mu)^{\alpha\beta}U_\mu^\dagger(n-\hat{\mu})\psi_\beta(n-\hat{\mu})\Big]\,. \end{aligned} \tag{6}$$

Highly ill-conditioned systems arise when we simulate close to criticality. In this model, this means when the physical fermion mass vanishes. Such a behavior is obtained when m_0 and β (bare mass and gauge coupling) take the values in Table 1, see Ref. [1]. We run simulations with parameters close to the ones in Table 1 for testing our method.

β	$-m_{\text{crit}}$
2	0.1968(9)
3	0.1351(2)
4	0.1033(1)
5	0.0840(1)
6	0.0719(1)

Table 1: Critical values for the Schwinger model [1].

Based on the ideas of Ref. [2], we implement a multilevel preconditioner to invert the Dirac matrix. The preconditioner is fed into the FGMRES method to solve the linear problem $Dx = \psi$. For smoothing the system we use *Schwarz Alternating Procedure* (SAP), a domain decomposition method. On the coarsest level the solver is GMRES (no preconditioning) with a tolerance of 10^{-2}. The construction of the interpolator P is done with an aggregation based approach, which we explain here. We use either a K- or V-cycling strategy.

2 Schwarz Alternating Procedure (SAP)

SAP is a domain decomposition method. Its general idea is based on dividing the lattice into blocks $\mathcal{L}_i$, each one containing a different set of variables $\mathcal{V}_i$. See for instance Figure 1. By restricting the application of the Dirac operator to these blocks, we can invert the local systems approximately and use them to update the global solution. We explain the implementation without discussing theoretical details, which are found in Refs. [2, 3].

We restrict the application of the Dirac matrix to the blocks with an operator

$$I_{\mathcal{V}} : \mathcal{V}_i \to \mathcal{V}, \tag{7}$$

where $\mathcal{V}$ corresponds to the whole set of variables and $\mathcal{V}_i \subset \mathcal{V}$ to a subset. We define the local operators

$$D_i := I_{\mathcal{V}_i}^T D I_{\mathcal{V}_i} \quad \text{and} \quad A_i := I_{\mathcal{V}_i} D_i^{-1} I_{\mathcal{V}_i}^T. \tag{8}$$

We use the solutions of the local systems

$$D_i e_i = I_{\mathcal{V}_i}^T r, \quad r = \psi - Dx \tag{9}$$

to update the global solution x. Since D only considers nearest-neighbour coupling, the blocks can be split into two sets forming a checkerboard pattern (red/black decomposition). In this manner, we

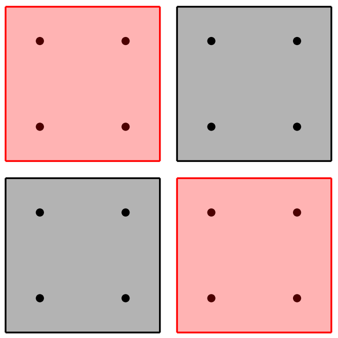


Figure 1: Red/Black decomposition of the lattice. Each point inside the blocks has two complex degrees of freedom.

can solve equations 9 simultaneously for all the red or black blocks. This allows for parallelization. We define one SAP iteration as

$$x \leftarrow (I - KD)x + K\psi, \tag{10}$$

where

$$K = A_{\text{black}}(I - DA_{\text{red}}) + A_{\text{red}}, \quad A_{\text{color}} = \sum_{i \in \text{color}} A_i. \tag{11}$$

In practice, SAP is implemented as we show in Algorithm 1. The restricted Dirac operator is the same as eq. (3), but only considering the sites that belong to the same block

$$\begin{aligned} \left(D_{\alpha\beta}[\,n, n'\,]\right)_{\mathcal{L}_i} &= (m_0 + 2)\delta^{\alpha\beta}\delta_{n,n'} \\ &\quad - \frac{1}{2}\sum_{\mu=0}^{1}\left[(1-\sigma_\mu)^{\alpha\beta}\, U_\mu(n)\delta_{n+\hat{\mu},n'} + (1+\sigma_\mu)^{\alpha\beta}\, U_\mu^\dagger(n-\hat{\mu})\delta_{n-\hat{\mu},n'}\right], \\ &\quad n, n', n+\hat{\mu}, n-\hat{\mu} \in \mathcal{L}_i. \end{aligned} \tag{12}$$

If one site is not in $\mathcal{L}_i$, we fix the corresponding term to zero. To invert the local systems we use GMRES to achieve a relative tolerance of 10^{-3}. This low-precision solution makes SAP a non-stationary method which is not guaranteed to converge as a stand-alone solver when the matrix is very ill-conditioned. For that reason, SAP is better used as a preconditioner or smoother. For DDαAMG, SAP is used as the latter for every grid level.

Algorithm 1 Schwarz Alternating Procedure

```
Inputs: $\psi$, $b$. Output: $\psi$
$r = \psi - D\psi$
for $i = 1, \cdots, \nu$:
  for all red $i$:
    $\psi \leftarrow \psi + A_i^{-1} r$
  $r = \psi - D\psi$
  for all black $i$:
    $\psi \leftarrow \psi + A_i^{-1} r$
  $r = \psi - D\psi$
```

3 Aggregation based approach

DDα is mainly characterized by the construction of its interpolator P. This follows an aggregation approach, which is based on the lattice geometry, and a set of *test vectors* $\{w_i | i = 1, \cdots, N_v\}$. These vectors should capture the *null-space* to improve the convergence properties of the method. They are obtained during a setup phase, which we describe in this section. The coarse grid matrix is computed with the Galerkin product $D_c = P^\dagger D P$.

To build the aggregates, we first decompose the lattice into N_B blocks. The exact definition for the blocking is in Definition 2.2 of Ref. [2], but for the sake of simplicity we explain the idea with an example. Let us consider a 6^2 volume. We call block_x and block_t to the number of sites contained in each block in the x and t directions respectively. We have the freedom to choose the value of block_i as long as N_i/block_i is an integer number. For instance, if block_x=2 and block_t=3 we have six lattice blocks

$$\begin{aligned}
\mathcal{L}_1 &= \{(0,0),(0,1),(0,2),(1,0),(1,1),(1,2)\},\\
\mathcal{L}_2 &= \{(0,3),(0,4),(0,5),(1,3),(1,4),(1,5)\},\\
\mathcal{L}_3 &= \{(2,0),(2,1),(2,2),(3,0),(3,1),(3,2)\},\\
\mathcal{L}_4 &= \{(2,3),(2,4),(2,5),(3,3),(3,4),(3,5)\},\\
\mathcal{L}_5 &= \{(4,0),(4,1),(4,2),(5,0),(5,1),(5,2)\},\\
\mathcal{L}_6 &= \{(4,3),(4,4),(4,5),(5,3),(5,4),(5,5)\}.
\end{aligned}$$

The ordered pairs are of the form (n_t, n_x). Other blockings can be built in a similar manner by taking cartesian products. In order to build the aggregates we also consider the spin component μ. For each lattice block we define two aggregates: one for $\mu = 0$ and another one for $\mu = 1$, that is

$$\mathcal{A}_{2i-1} = \mathcal{L}_i \times \{0\}, \quad \mathcal{A}_{2i} = \mathcal{L}_i \times \{1\}. \tag{13}$$

Thus, the number of aggregates is twice the number of lattice blocks, *i.e.* $N_\mathcal{A} = 2N_B$. The blocking used for the aggregates does not need to coincide with the blocking for SAP.

We now need a set of test vectors to build the interpolator. We describe in a merely practical way how to obtain them:

- Generate a set of N_v random vectors of dimension $2N_xN_t$. We denote the vectors as w_i.
- Approximately solve $Dx = 0$ with initial solution w_i for every test vector. Update $w_i \leftarrow x$. A couple of iterations with the smoother is enough. The idea is to make the test vectors rich in near-kernel components.
- Split the test vectors over the aggregates. We explain the idea. Let us say we have $N_\mathcal{A}$ aggregates. We restrict w_i by leaving those components that belong to an aggregate $\mathcal{A}$ unchanged, while zeroing the others. This way we create another $N_vN_\mathcal{A}$ vectors of the form

$$\left(w_i^{(j)}\right)_k = \begin{cases} (w_i)_k & \text{if } k \in \mathcal{A}_j, \\ 0 & \text{else,} \end{cases} \tag{14}$$

  where $j = 0, \ldots, N_\mathcal{A} - 1$, $i = 0, \ldots N_v - 1$ and $k = 0, \ldots 2N_xN_t - 1$. Afterwards, the new vectors are orthonormalized. We do the orthonormalization only among those vectors that share the same index j, *i.e.* only among those vectors restricted to the same aggregate. The interpolator is built by arranging the chopped vectors in columns

$$P = \left(w_1^{(1)} \middle| w_2^{(1)} \middle| \cdots \middle| w_{N_v}^{(1)} \middle| \cdots \middle| w_1^{(N_\mathcal{A})} \middle| w_2^{(N_\mathcal{A})} \middle| \cdots \middle| w_{N_v}^{(N_\mathcal{A})} \right). \tag{15}$$

It the operator is assembled correctly, the sets

$$V_j = \{w_1^{(j)}, w_2^{(j)}, \ldots, w_{N_v}^{(j)}\}, \quad j = 1, \ldots, N_\mathcal{A} \tag{16}$$

should be orthonormal.

- The quality of the interpolator is further improved with *bootstrap adaptivity.* That is, we use the current multilevel method defined by P to approximately solve $Dx = (w_i - Dw_i)$ and update $w_i \leftarrow x$. The resultant test vectors are orthonormalized again as in the previous step.

With these ingredients we can implement the multilevel method. Still, it is convenient to find an explicit representation for the coarse grid operator D_c to avoid calling P and $P^\dagger$ several times during the computation. We show how to do that in the next section. We introduce a new set of coarse variables which allow us to rewrite D_c in an analogue way as the original Dirac matrix.

For more than two levels we repeat the same steps: aggregate the coarse lattice and generate a set of test vectors which are later orthonormalized.

4 Constructing the coarse grid operator

The Dirac operator on the fine grid, eq. (3), is defined by the gauge links and considering nearest-neighbor couplings. On coarser levels, we can preserve a similar structure by introducing a set of *coarse gauge links* in such a way that

$$(D_c)_{xy} = [(m_0 + 2) - A(x)]\, \delta_{xy} - \sum_{\mu=0}^{1} \left(B_\mu(x)\delta_{x+\hat{\mu},y} + C_\mu(x)\delta_{x-\hat{\mu},y}\right), \tag{17}$$

where x, y are lattice blocks and A, B_μ, C_μ represent the coarse links. On the coarse grid, each point on the lattice acquires $2N_v$ degrees of freedom. We show how to find and explicit representation for $A(x), B_\mu(x)$ and $C_\mu(x)$ for the two-level case. Later we show the generalization for the multilevel method.

We first write the components of P

$$P_{ij} = [w_c^{(x,\alpha)}]_i, \tag{18}$$

where c, α and x represent the test vector, spin and lattice block respectively

$$c = 0, \ldots, N_v - 1, \quad \alpha = 0, 1, \quad x = 0, \ldots, N_B - 1. \tag{19}$$

We can relate these numbers with the vectorized index j through

$$c = j \bmod N_v, \quad a = \mathrm{int}(j/N_v), \quad x = \mathrm{int}(a/2), \quad \alpha = a \bmod 2, \tag{20}$$

where a is the aggregate associated to j. On the other hand, the vectorized index i represents a variable on the fine grid (n, α) (lattice site and spin). This allows us to write

$$P_{ij} = \left[w_c^{(x,\alpha)}\right]_\alpha (n), \quad i = 2n + \alpha, \quad j = N_v a + c, \tag{21}$$

where $i = 0, \dots, 2N_xN_t - 1$ and $j = 0, \dots, N_{\mathcal{A}}N_v - 1$. In a similar manner, we have

$$P^\dagger_{ji} = \left[w_c^{*(x,\alpha)}\right]_\alpha(n), \quad i = 2n+\alpha, \quad j = N_v a + c. \tag{22}$$

Since we split the test vectors over the aggregates, the following is satisfied

$$\text{if} \quad (n,\alpha) \notin \mathcal{A}_{(x,\alpha)} \quad \Longrightarrow \quad \left[w_c^{(x,\alpha)}\right]_\alpha(n) = 0, \tag{23}$$

where (x,α) defines the aggregate $\mathcal{A}_{(x,\alpha)} = x \times \{\alpha\}$.

Let us find an expression for $(P^\dagger D P)_{ij}$ with this information

$$\begin{aligned}(D_c)_{ij} &= (P^\dagger D P)_{ij} \quad i,j = 0,\dots,N_{\mathcal{A}}N_v - 1 \\ &= \sum_{kl} P^\dagger_{ik} D_{kl} P_{lj} = \sum_{kl} P^\dagger_{ik} D_{kl}\left[w_b^{(y,\beta)}\right]_l = \sum_k P^\dagger_{ik}\left(D\, w_b^{(y,\beta)}\right)_k \end{aligned} \tag{24}$$

The indices y, β and b are related to j according to eq. (20) and l represents a variable on the fine grid. We use eq. (6) to simplify the application of D over $w_b^{(y,\beta)}$

$$\begin{aligned}\left[D\, w_b^{(y,\beta)}\right]_\gamma(n) &= (m_0+2)\left[w_b^{(y,\beta)}\right]_\gamma(n) \\ &- \frac{1}{2}\sum_{\mu,\delta=0}^{1}\left[(1-\sigma_\mu)^{\gamma\delta}\, U_\mu(n)\left[w_b^{(y,\beta)}\right]_\delta(n+\hat{\mu}) + (1+\sigma_\mu)^{\gamma\delta}\, U^\dagger_\mu(n-\hat{\mu})\left[w_b^{(y,\beta)}\right]_\delta(n-\hat{\mu})\right].\end{aligned}$$

If we consider that $k = 2\gamma + n$, we can write eq. (24) as

$$\begin{aligned}[D_c]^{\alpha\beta}_{cb}(x,y) &= \sum_{n\in\Lambda,\,\gamma}(m_0+2)\left[w_c^{*(x,\alpha)}\right]_\gamma(n)\left[w_b^{(y,\beta)}\right]_\gamma(n) \\ &- \frac{1}{2}\sum_{n\in\Lambda,\mu,\gamma,\delta=0}^{1}\left[w_c^{*(x,\alpha)}\right]_\gamma(n)\left[(1-\sigma_\mu)^{\gamma\delta}\, U_\mu(n)\left[w_b^{(y,\beta)}\right]_\delta(n+\hat{\mu}) + (1+\sigma_\mu)^{\gamma\delta}\, U^\dagger_\mu(n-\hat{\mu})\left[w_b^{(y,\beta)}\right]_\delta(n-\hat{\mu})\right].\end{aligned}$$

We relate this equation with eq. (24) through

$$\begin{aligned} c &= i \bmod N_v, \quad a = \mathrm{int}(i/N_v), \quad x = \mathrm{int}(a/2), \quad \alpha = a \bmod 2, \\ b &= j \bmod N_v, \quad a' = \mathrm{int}(j/N_v), \quad y = \mathrm{int}(a'/2), \quad \beta = a' \bmod 2, \end{aligned} \tag{25}$$

Due to the local orthonormalization of the test vectors, the first term is different from zero when $x = y, \alpha = \beta$ and $c = b$. For the second term, the test vectors are zero unless $\alpha = \gamma$ and $\beta = \delta$. Thus,

$$\begin{aligned}[D_c]^{\alpha\beta}_{cb}(x,y) &= (m_0+2)\delta_{xy}\delta^{\alpha\beta}\delta_{cb} \\ &- \frac{1}{2}\sum_{n\in\Lambda,\mu=0}^{1}\left[w_c^{*(x,\alpha)}\right]_\alpha(n)\left[(1-\sigma_\mu)^{\alpha\beta}\, U_\mu(n)\left[w_b^{(y,\beta)}\right]_\beta(n+\hat{\mu}) + (1+\sigma_\mu)^{\alpha\beta}\, U^\dagger_\mu(n-\hat{\mu})\left[w_b^{(y,\beta)}\right]_\beta(n-\hat{\mu})\right].\end{aligned}$$

The lattice blocking implies that the second term is only different from zero when $y = x$, $y = x + \hat{\mu}$

or $y = x - \hat{\mu}$. Then, we have

$$
\begin{aligned}
[D_c]^{\alpha\beta}_{cb}(x,y) = (m_0+2)\delta_{xy}\delta^{\alpha\beta}\delta_{cb} &- \frac{1}{2}\sum_{\mu,n\in x}(1-\sigma_\mu)^{\alpha\beta}\left[w_c^{*(x,\alpha)}\right]_\alpha(n)\,U_\mu(n)\,\left[w_b^{(x,\beta)}\right]_\beta(n+\hat{\mu}) \\
&- \frac{1}{2}\sum_{\mu,n\in x}(1-\sigma_\mu)^{\alpha\beta}\left[w_c^{*(x,\alpha)}\right]_\alpha(n)\,U_\mu(n)\,\left[w_b^{(x+\hat{\mu},\beta)}\right]_\beta(n+\hat{\mu}) \\
&- \frac{1}{2}\sum_{\mu,n\in x}(1+\sigma_\mu)^{\alpha\beta}\left[w_c^{*(x,\alpha)}\right]_\alpha(n)\,U^\dagger_\mu(n-\hat{\mu})\,\left[w_b^{(x,\beta)}\right]_\beta(n-\hat{\mu}) \\
&- \frac{1}{2}\sum_{\mu,n\in x}(1+\sigma_\mu)^{\alpha\beta}\left[w_c^{*(x,\alpha)}\right]_\alpha(n)\,U^\dagger_\mu(n-\hat{\mu})\,\left[w_b^{(x-\hat{\mu},\beta)}\right]_\beta(n-\hat{\mu}). \qquad (26)
\end{aligned}
$$

Let us define

$$
\begin{aligned}
[A(x)]^{\alpha\beta}_{cb} &:= \frac{1}{2}\sum_{\mu,n\in x}\Bigg[(1-\sigma_\mu)^{\alpha\beta}\left[w_c^{*(x,\alpha)}\right]_\alpha(n)\,U_\mu(n)\,\left[w_b^{(x,\beta)}\right]_\beta(n+\hat{\mu}) \\
&\quad + (1+\sigma_\mu)^{\alpha\beta}\left[w_c^{*(x,\alpha)}\right]_\alpha(n)\,U^\dagger_\mu(n-\hat{\mu})\,\left[w_b^{(x,\beta)}\right]_\beta(n+\hat{\mu})\Bigg] \qquad (27) \\
[B_\mu(x)]^{\alpha\beta}_{cb} &:= \frac{1}{2}\sum_{n\in x}(1-\sigma_\mu)^{\alpha\beta}\left[w_c^{*(x,\alpha)}\right]_\alpha(n)\,U_\mu(n)\,\left[w_b^{(x+\hat{\mu},\beta)}\right]_\beta(n+\hat{\mu}) \qquad (28) \\
[C_\mu(x)]^{\alpha\beta}_{cb} &:= \frac{1}{2}\sum_{n\in x}(1+\sigma_\mu)^{\alpha\beta}\left[w_c^{*(x,\alpha)}\right]_\alpha(n)\,U^\dagger_\mu(n-\hat{\mu})\,\left[w_b^{(x-\hat{\mu},\beta)}\right]_\beta(n-\hat{\mu}). \qquad (29)
\end{aligned}
$$

Thus, if we have a vector v on the coarse grid, we can write the application $D_c v$ as

$$
\begin{aligned}
(D_c\,v)^\alpha_c(x) = (m_0+2)v_c^{(x,\alpha)} &- \sum_{y,\beta,b}[A(x)]^{\alpha\beta}_{cb}\;\delta_{xy}\,v_b^{(y,\beta)} \\
&- \sum_\mu\left([B_\mu(x)]^{\alpha\beta}_{cb}\;\delta_{x+\hat{\mu},y}\,v_b^{(y,\beta)} + [C_\mu(x)]^{\alpha\beta}_{cb}\;\delta_{x-\hat{\mu},y}\,v_b^{(y,\beta)}\right). \qquad (30)
\end{aligned}
$$

Therefore

$$
(D_c)_{xy} = [(m_0+2) - A(x)]\,\delta_{xy} - \sum_{\mu=0}^{1}\left[B_\mu(x)\,\delta_{x+\hat{\mu},y} + C_\mu(x)\,\delta_{x-\hat{\mu},y}\right], \qquad (31)
$$

where we assume that a sum over α,β,c,b is performed when multiplied by a vector on the coarse grid. We only build A, B_μ and C_μ once and store them in memory. For the expression above to work, we need to first locally orthonormalize the test vectors.

Now, let us say we want to proceed to the next coarse level. We again have to split the lattice into $\hat{N}_B$ blocks (the previous blocks are now our lattice sites), build $\hat{N}_{\mathcal{A}} = 2\hat{N}_B$ aggregates and a set of $\hat{N}_v$ test vectors which are locally orthonormalized. The components of the interpolator are

$$
\hat{P}_{ij} = \left[w_{\hat{c}}^{(\hat{x},\hat{\alpha})}\right]_{c\alpha}(x). \qquad (32)
$$

where

$$
\begin{gathered}
i = 0,\ldots,N_{\mathcal{A}}N_v - 1, \quad j = 0,\ldots,\hat{N}_{\mathcal{A}}\hat{N}_v - 1, \\
c = 0,\ldots,N_v-1, \quad \alpha = 0,1, \quad x = 0,\ldots,N_B-1, \\
\hat{c} = 0,\ldots,\hat{N}_v-1, \quad \hat{\alpha} = 0,1, \quad \hat{x} = 0,\ldots,\hat{N}_B-1. \qquad (33)
\end{gathered}
$$

To build $\hat{D}_c = \hat{P}^\dagger D_c P$ we do a similar procedure as before

$$\begin{aligned}
(\hat{D}_c)_{ij} = \left[\hat{D}_c\right]^{\hat{\alpha}\hat{\beta}}_{\hat{c}\hat{b}}(\hat{x},\hat{y}) &= \sum_{x,c,b,\alpha,\beta} (m_0+2)\left[w^{*(\hat{x},\hat{\alpha})}_{\hat{c}}\right]_{c\alpha}(x)\left[w^{(\hat{y},\hat{\beta})}_{\hat{b}}\right]_{c\alpha}(x) \\
&- \sum_{x,c,b,\alpha,\beta}\left[w^{*(\hat{x},\hat{\alpha})}_{\hat{c}}\right]_{c\alpha}(x)\,[A(x)]^{\alpha\beta}_{cb}\left[w^{(\hat{y},\hat{\beta})}_{\hat{b}}\right]_{c\alpha}(x) \\
&- \sum_{\mu,x,c,b,\alpha,\beta}\left[w^{*(\hat{x},\hat{\alpha})}_{\hat{c}}\right]_{c\alpha}(x)\,[B_\mu(x)]^{\alpha\beta}_{cb}\left[w^{(\hat{y},\hat{\beta})}_{\hat{b}}\right]_{c\alpha}(x+\hat{\mu}) \\
&- \sum_{\mu,x,c,b,\alpha,\beta}\left[w^{*(\hat{x},\hat{\alpha})}_{\hat{c}}\right]_{c\alpha}(x)\,[C_\mu(x)]^{\alpha\beta}_{cb}\left[w^{(\hat{y},\hat{\beta})}_{\hat{b}}\right]_{c\alpha}(x-\hat{\mu}). \qquad (34)
\end{aligned}$$

The indices i, j are a vectorization of $(\hat{c}, \hat{\alpha}, \hat{x})$ and $(\hat{b}, \hat{\beta}, \hat{y})$ respectively. Remember that

$$\left[w^{(\hat{x},\hat{\alpha})}_{\hat{b}}\right]_{c\alpha}(x) = 0 \quad \text{if} \quad (x,\alpha,c) \notin \mathcal{A}_{(\hat{y},\hat{\beta})}. \qquad (35)$$

This, together with the local orthonormalization of the test vectors, imply that the first term is again just $m_0 + 2$ multiplied by deltas. We should also note that unless $\hat{\alpha} = \alpha$ and $\hat{\beta} = \beta$, the test vector components are zero, so we can skip the sums over α, β. In addition, the nearest neighbor coupling and the lattice blocking altogether imply

$$\begin{aligned}
\left[\hat{D}_c\right]^{\alpha\beta}_{\hat{c}\hat{b}}(\hat{x},\hat{y}) = (m_0+2)\delta_{\hat{x}\hat{y}}\delta^{\alpha\beta}\delta_{\hat{c}\hat{b}} &- \sum_{x\in\hat{x},c,b}\left[w^{*(\hat{x},\alpha)}_{\hat{c}}\right]_{c\alpha}(x)\,[A(x)]^{\alpha\beta}_{cb}\left[w^{(\hat{x},\beta)}_{\hat{b}}\right]_{b\beta}(x)\,\delta_{\hat{x}\hat{y}} \\
&- \sum_{\mu,x\in\hat{x},c,b}\left[w^{*(\hat{x},\alpha)}_{\hat{c}}\right]_{c\alpha}(x)\,[B_\mu(x)]^{\alpha\beta}_{cb}\left[w^{(\hat{x},\beta)}_{\hat{b}}\right]_{b\beta}(x+\hat{\mu})\,\delta_{\hat{x}\hat{y}} \\
&- \sum_{\mu,x\in\hat{x},c,b}\left[w^{*(\hat{x},\alpha)}_{\hat{c}}\right]_{c\alpha}(x)\,[C_\mu(x)]^{\alpha\beta}_{cb}\left[w^{(\hat{x},\beta)}_{\hat{b}}\right]_{b\beta}(x-\hat{\mu})\,\delta_{\hat{x}\hat{y}} \\
&- \sum_{\mu,x\in\hat{x},c,b}\left[w^{*(\hat{x},\alpha)}_{\hat{c}}\right]_{c\alpha}(x)\,[B_\mu(x)]^{\alpha\beta}_{cb}\left[w^{(\hat{x}+\hat{\mu},\beta)}_{\hat{b}}\right]_{b\beta}(x+\hat{\mu})\,\delta_{\hat{x}+\hat{\mu},\hat{y}} \\
&- \sum_{\mu,x\in\hat{x},c,b}\left[w^{*(\hat{x},\alpha)}_{\hat{c}}\right]_{c\alpha}(x)\,[C_\mu(x)]^{\alpha\beta}_{cb}\left[w^{(\hat{x}-\hat{\mu},\beta)}_{\hat{b}}\right]_{b\beta}(x-\hat{\mu})\,\delta_{\hat{x}-\hat{\mu},\hat{y}}. \qquad (36)
\end{aligned}$$

We define

$$\begin{aligned}
\left[\hat{A}(\hat{x})\right]^{\alpha\beta}_{\hat{c}\hat{b}} &:= \sum_{x\in\hat{x},c,b}\left[w^{*(\hat{x},\alpha)}_{\hat{c}}\right]_{c\alpha}(x)\,[A(x)]^{\alpha\beta}_{cb}\left[w^{(\hat{x},\beta)}_{\hat{b}}\right]_{b\beta}(x) \\
&+ \sum_{\mu,x\in\hat{x},c,b}\left[w^{*(\hat{x},\alpha)}_{\hat{c}}\right]_{c\alpha}(x)\,[B_\mu(x)]^{\alpha\beta}_{cb}\left[w^{(\hat{x},\beta)}_{\hat{b}}\right]_{b\beta}(x+\hat{\mu}) \\
&+ \sum_{\mu,x\in\hat{x},c,b}\left[w^{*(\hat{x},\alpha)}_{\hat{c}}\right]_{c\alpha}(x)\,[C_\mu(x)]^{\alpha\beta}_{cb}\left[w^{(\hat{x},\beta)}_{\hat{b}}\right]_{b\beta}(x-\hat{\mu}), \\
\left[\hat{B}_\mu(\hat{x})\right]^{\alpha\beta}_{\hat{c}\hat{b}} &:= \sum_{\mu,x\in\hat{x},c,b}\left[w^{*(\hat{x},\alpha)}_{\hat{c}}\right]_{c\alpha}(x)\,[B_\mu(x)]^{\alpha\beta}_{cb}\left[w^{(\hat{x}+\hat{\mu},\beta)}_{\hat{b}}\right]_{b\beta}(x+\hat{\mu}), \\
\left[\hat{C}_\mu(\hat{x})\right]^{\alpha\beta}_{\hat{c}\hat{b}} &:= \sum_{\mu,x\in\hat{x},c,b}\left[w^{*(\hat{x},\alpha)}_{\hat{c}}\right]_{c\alpha}(x)\,[C_\mu(x)]^{\alpha\beta}_{cb}\left[w^{(\hat{x}-\hat{\mu},\beta)}_{\hat{b}}\right]_{b\beta}(x-\hat{\mu}). \qquad (37)
\end{aligned}$$

Thus, the application of $\hat{D}_c$ on a coarse vector $\hat{v}$ is

$$(\hat{D}_c\, v)_{\hat{c}\alpha}(\hat{x}) = (m_0+2)v_{\hat{c}}^{(\hat{x},\alpha)} - \sum_{\hat{y},\beta,\hat{b}} \left[\hat{A}(\hat{x})\right]_{\hat{c}\hat{b}}^{\alpha\beta} \delta_{\hat{x}\hat{y}}\, v_{\hat{b}}^{(\hat{y},\beta)}$$
$$- \sum_{\mu}\left(\left[\hat{B}_\mu(\hat{x})\right]_{\hat{c}\hat{b}}^{\alpha\beta} \delta_{\hat{x}+\hat{\mu},\hat{y}}\, v_{\hat{b}}^{(\hat{y},\beta)} + \left[\hat{C}_\mu(\hat{x})\right]_{\hat{c}\hat{b}}^{\alpha\beta} \delta_{\hat{x}-\hat{\mu},\hat{y}}\, v_{\hat{b}}^{(\hat{y},\beta)} \right). \quad (38)$$

Therefore, the operator $\hat{D}_c$ also has the form

$$(\hat{D}_c)_{\hat{x}\hat{y}} = \left[(m_0+2) - \hat{A}(\hat{x})\right]\delta_{\hat{x}\hat{y}} - \sum_{\mu=0}^{1}\left[\hat{B}_\mu(\hat{x})\delta_{\hat{x}+\hat{\mu},\hat{y}} + \hat{C}_\mu(\hat{x})\delta_{\hat{x}-\hat{\mu},\hat{y}}\right]. \quad (39)$$

Clearly, we can repeat the same process to go to the next level and we will again find the same form for the operator, with the same definition for the coarse links in eq. (37). In fact, we can even use the same expression for the finest level if we define

$$[A(x)]_{cb}^{\alpha\beta} := 0,\ [B_\mu(x)]_{cb}^{\alpha\beta} := \frac{1}{2}(1-\sigma_\mu)^{\alpha\beta}U_\mu(x),\ [C_\mu(x)]_{cb}^{\alpha\beta} := \frac{1}{2}(1+\sigma_\mu)^{\alpha\beta}U_\mu^\dagger(x-\hat{\mu}). \quad (40)$$

In that case c and b are spurious. This way, we can always use eq. (37) to form the coarse gauge links to proceed to the next multigrid level. As a final note, the minus sign of the anti-periodic boundary conditions is taken into account in $\delta_{x+\hat{\mu},\hat{y}}$ and $\delta_{x-\hat{\mu},\hat{y}}$, not in the coarse links A, B_μ, C_μ.

5 Results

We generated configurations of the two degenerate flavor Schwinger model with a Hybrid Monte Carlo algorithm (HMC) for $V = 64^2, 128^2$ and 256^2. We used $\beta = 2$ and $m_0 = -0.1884$, which is close to the critical mass $m_0 = -0.1968(9)$ reported in Ref. [1]. We use our DDαAMG with 2, 3 and 4 levels to invert the matrix and compare the performance. We compare both K- and V-cycles as well. For every level we use a fixed number of test vectors. We tried $N_v = 10$ and 20. We randomly generated the right-hand side with a uniform distribution and it is always the same for each volume. Extensive testing showed that the bootstrap adaptivity does not really improve the convergence properties for this two-dimensional model. Instead, we only smooth the test vectors with 4 SAP iterations by approximating $Dx = 0$ and setting $w_i \leftarrow x$. On the fine grid, the initial solution is randomly generated. For the multilevel method, the initial solution is the restriction of the test vectors from the previous level.

From Figure 2 to Figure 7, we show the evolution of the residual according to the number of iterations in different scenarios. We use the data from one conf and one right-hand side for a simple comparison. The captions are self-explanatory. In Table 2, we display the parameters we choose for the method. From Table 3 to 5, we display the lattice blocking used for the aggregation and for SAP.

In figures 8, 9, 10 and 11 we show the average number of iterations it takes for the method to converge when applying bootstrap adaptivity. We average over 10 runs. For each one, we randomly generate a new set of test vectors which are then improved in the set up phase. The configurations and right-hand sides are still the same from the previous figures. We also show the execution time, although the code is not fully optimized. In many cases, less iterations does not imply less execution time.

Pre-smoothing	0
Post-smoothing	2
GMRES for SAP restart length	5
GMRES for SAP restarts	5
GMRES for SAP relative tolerance	10^{-3}
GMRES for coarsest level restart length	20
GMRES for coarsest level restarts	20
GMRES for coarsest level relative tolerance	10^{-1}
Outer FGMRES restart length	20
Outer FGMRES restarts	50
Outer FGMRES relative tolerance	10^{-10}

Table 2: DDαAMG parameters.

Levels	block_x	block_t	SAP block_x	SAP block_t
0	4	4	4	4

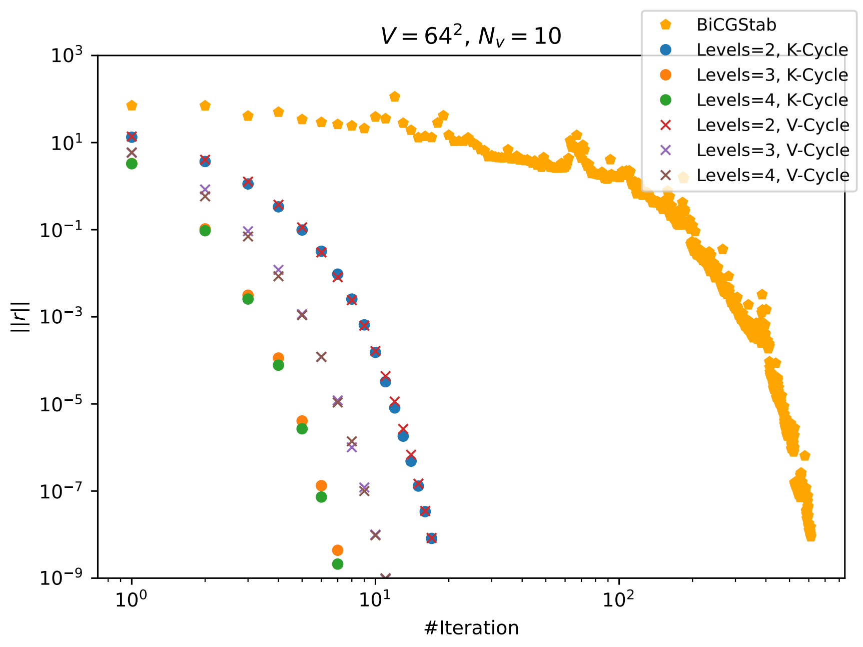
Table 3: Two levels blocking.

Levels	block_x	block_t	SAP block_x	SAP block_t
0	8	8	4	4
1	4	4	4	4

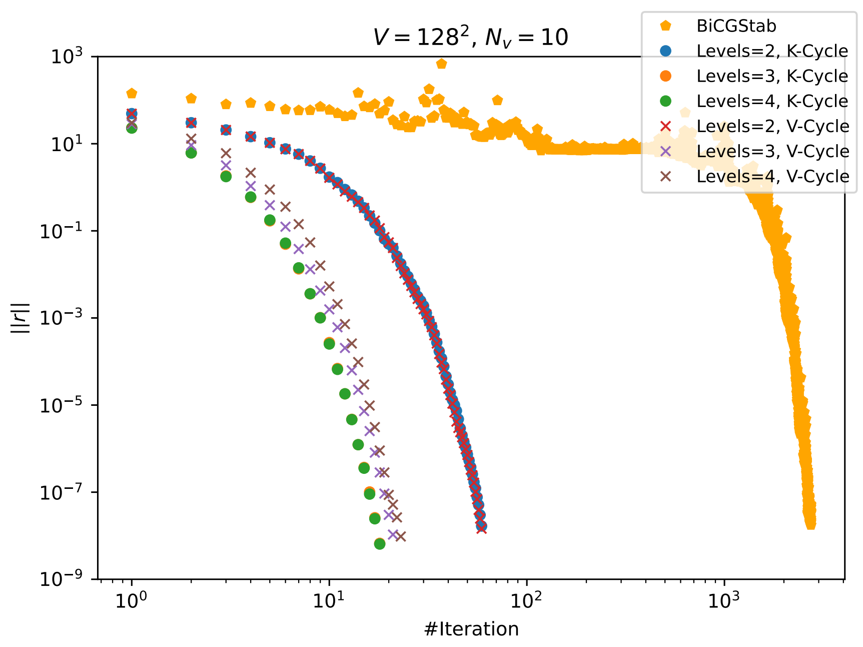
Table 4: Three levels blocking.

Levels	block_x	block_t	SAP block_x	SAP block_t
0	8	8	4	4
1	4	4	4	4
2	2	2	2	2

Table 5: Four levels blocking.



$V = 64^2$, $N_v = 10$
BiCGStab
Levels=2, K-Cycle
Levels=3, K-Cycle
Levels=4, K-Cycle
Levels=2, V-Cycle
Levels=3, V-Cycle
Levels=4, V-Cycle
$\|r\|$
#Iteration



$V = 128^2$, $N_v = 10$
BiCGStab
Levels=2, K-Cycle
Levels=3, K-Cycle
Levels=4, K-Cycle
Levels=2, V-Cycle
Levels=3, V-Cycle
Levels=4, V-Cycle
$\|r\|$
#Iteration

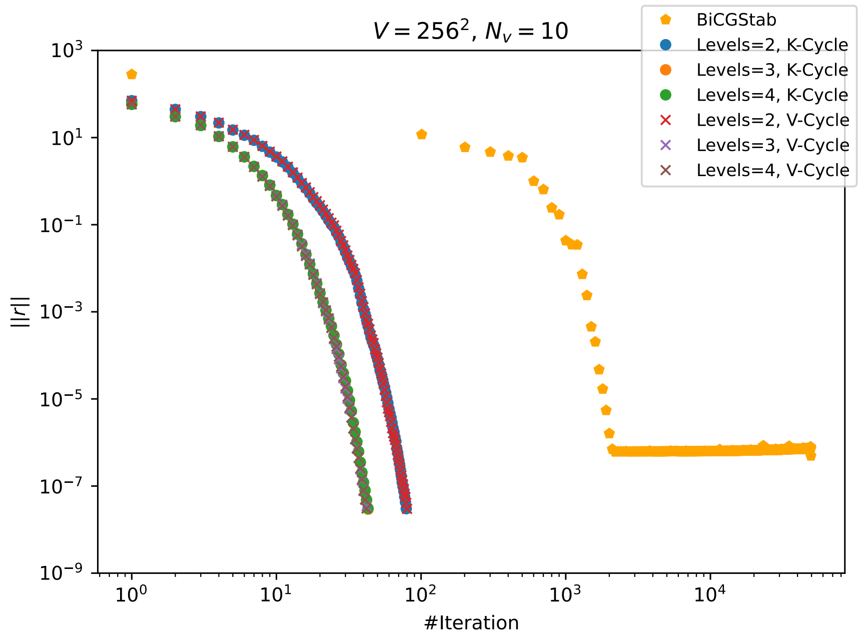
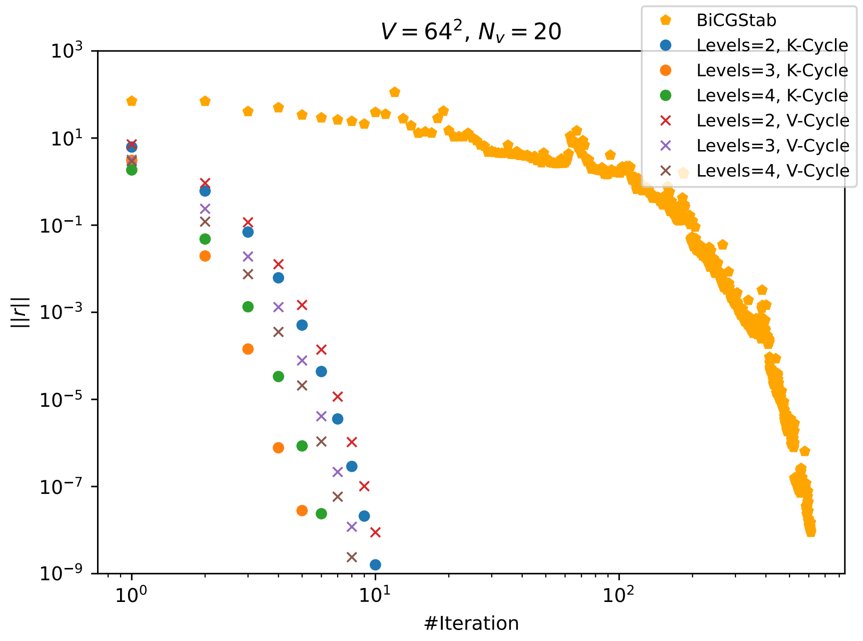
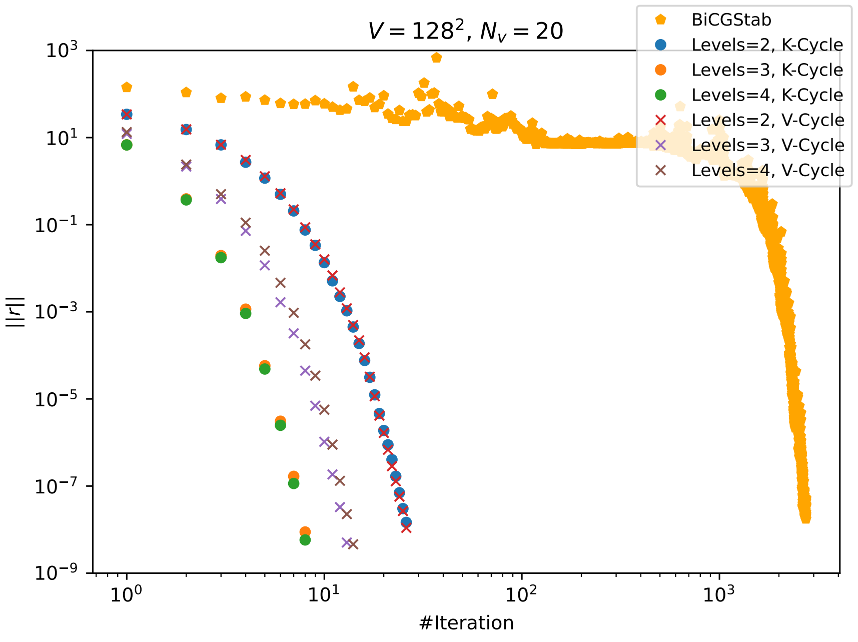


Figure 2: Residual evolution. We compare the 2, 3 and 4 level methods for a fixed volume, number of test vectors $N_v = 10$, and both K- and V-cycles.





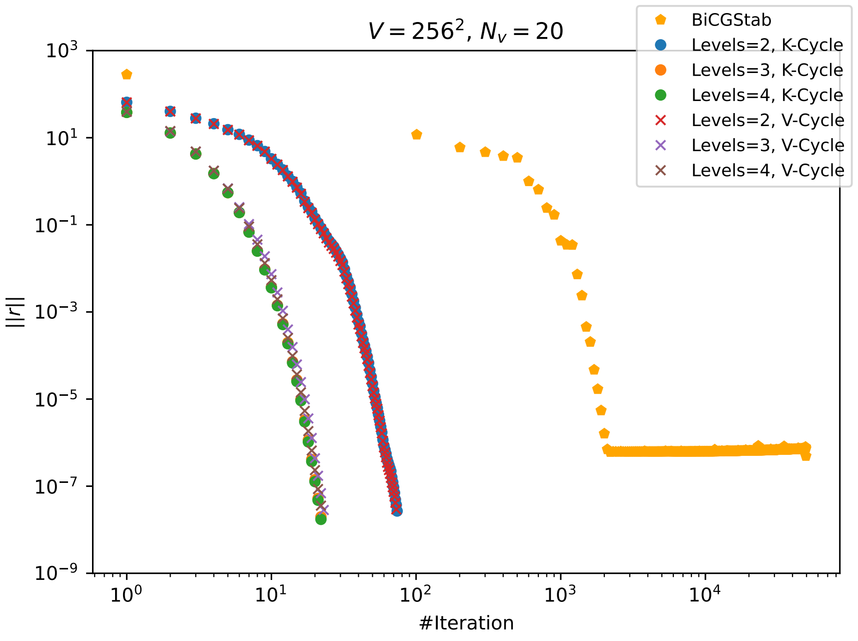
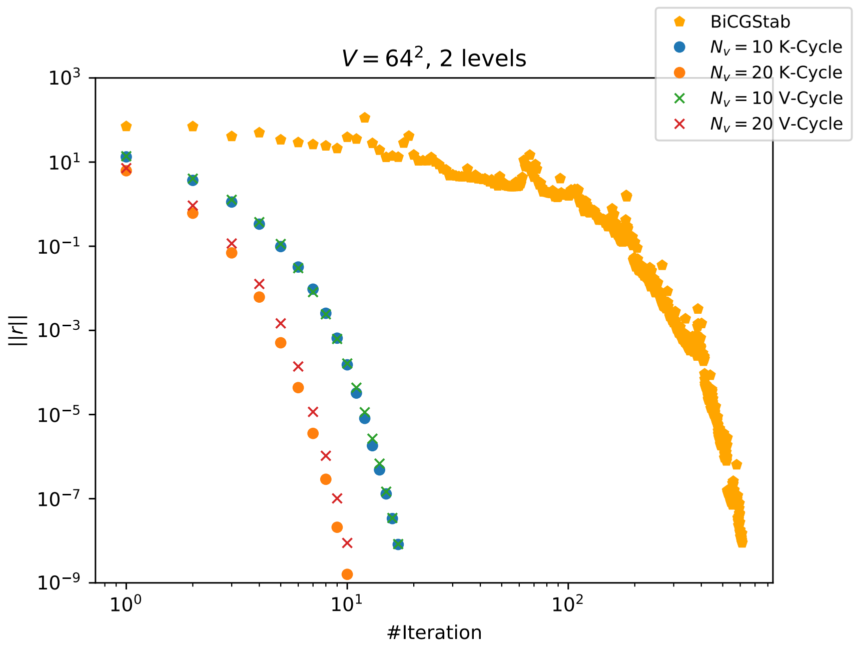
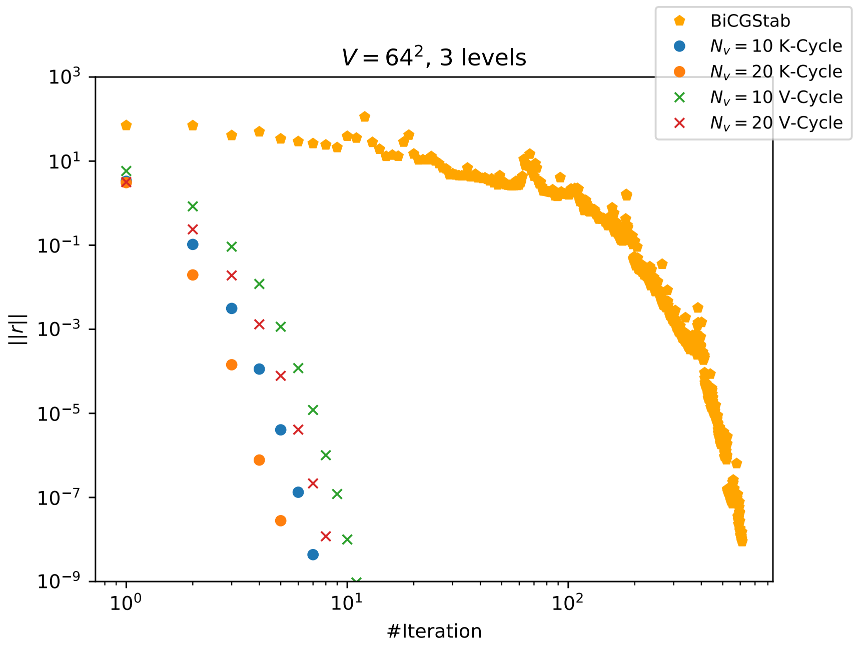


Figure 3: Residual evolution. We compare the 2, 3 and 4 level methods for a fixed volume, number of test vectors $N_v = 20$, and both K- and V-cycles.


$V = 64^2$, 2 levels
BiCGStab
$N_v = 10$ K-Cycle
$N_v = 20$ K-Cycle
$N_v = 10$ V-Cycle
$N_v = 20$ V-Cycle
$||r||$
#Iteration


$V = 64^2$, 3 levels
BiCGStab
$N_v = 10$ K-Cycle
$N_v = 20$ K-Cycle
$N_v = 10$ V-Cycle
$N_v = 20$ V-Cycle
$||r||$
#Iteration

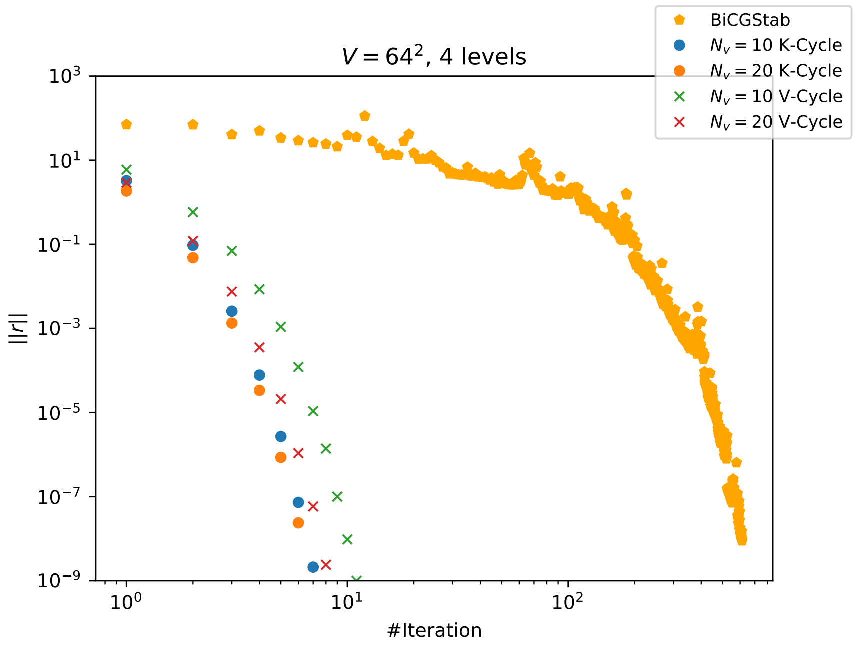
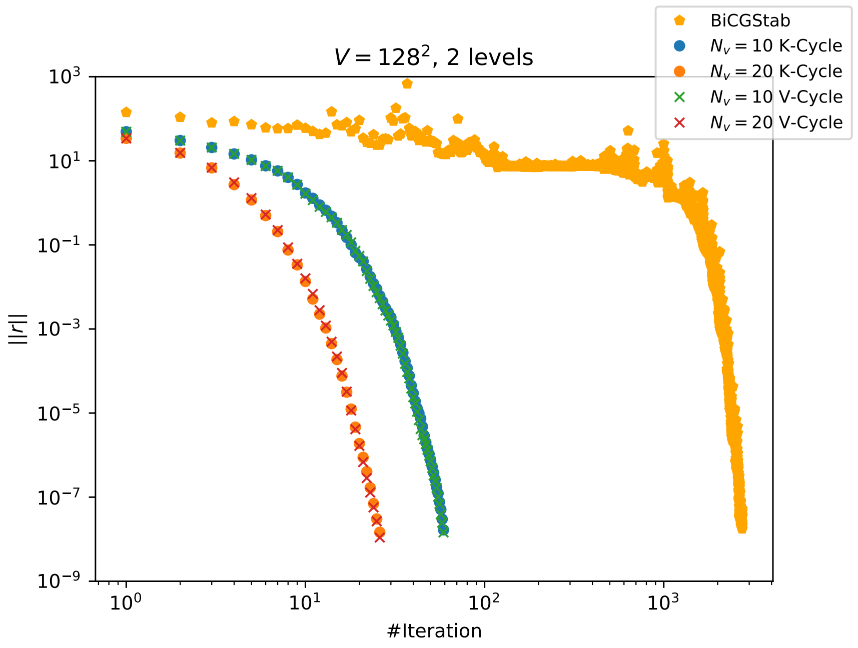
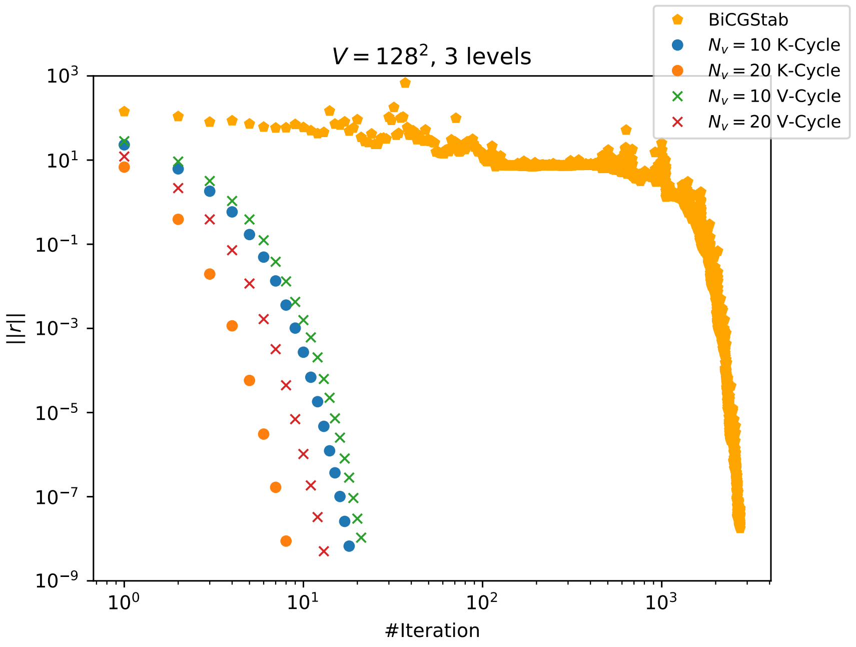


Figure 4: Solution with different levels, $N_v = 10, 20$ and both cycles. We fix the volume to $V = 64^2$.





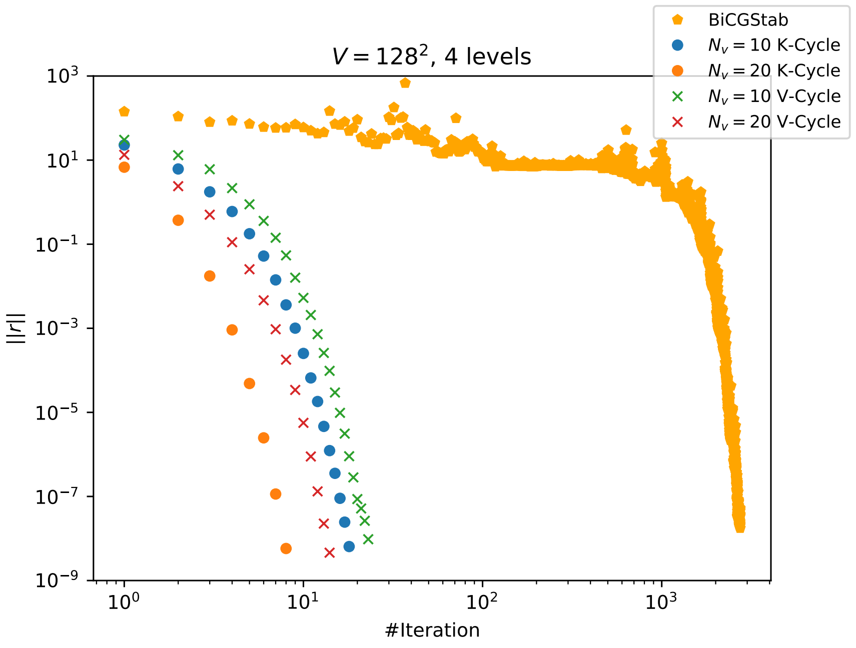
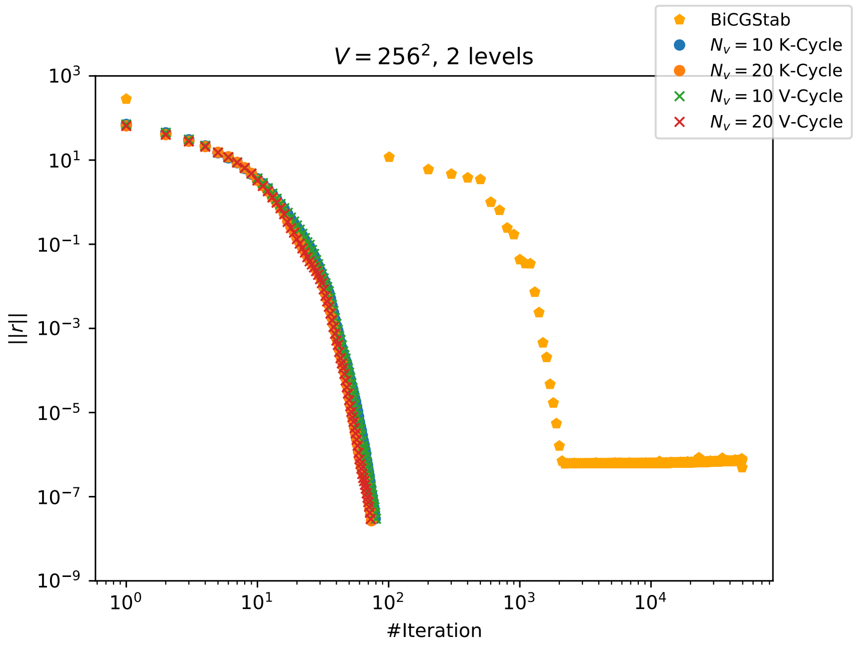
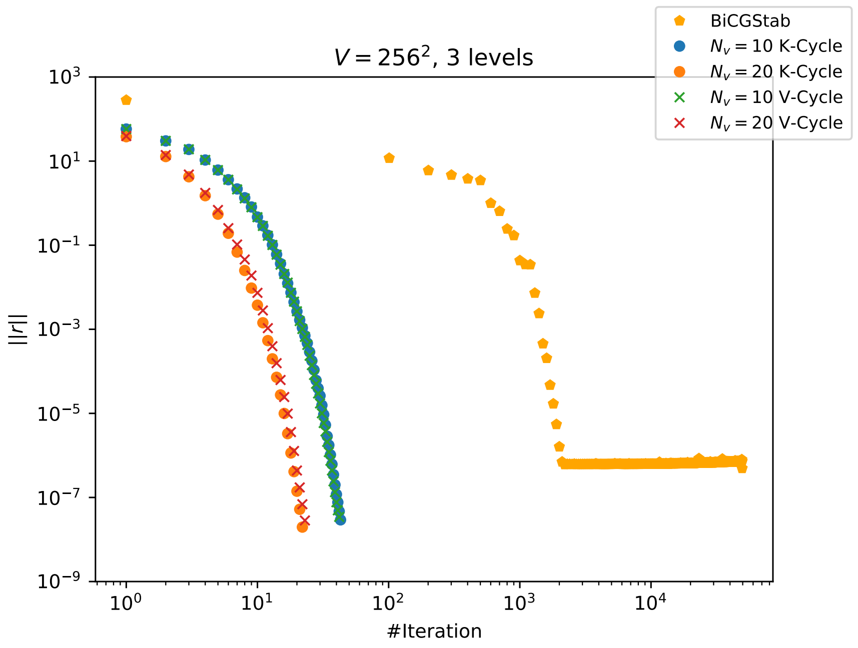


Figure 5: Solution with different levels, $N_v = 10, 20$ and both cycles. We fix the volume to $V = 128^2$.


$V = 256^2$, 2 levels
BiCGStab
$N_v = 10$ K-Cycle
$N_v = 20$ K-Cycle
$N_v = 10$ V-Cycle
$N_v = 20$ V-Cycle
$||r||$
#Iteration


$V = 256^2$, 3 levels
BiCGStab
$N_v = 10$ K-Cycle
$N_v = 20$ K-Cycle
$N_v = 10$ V-Cycle
$N_v = 20$ V-Cycle
$||r||$
#Iteration

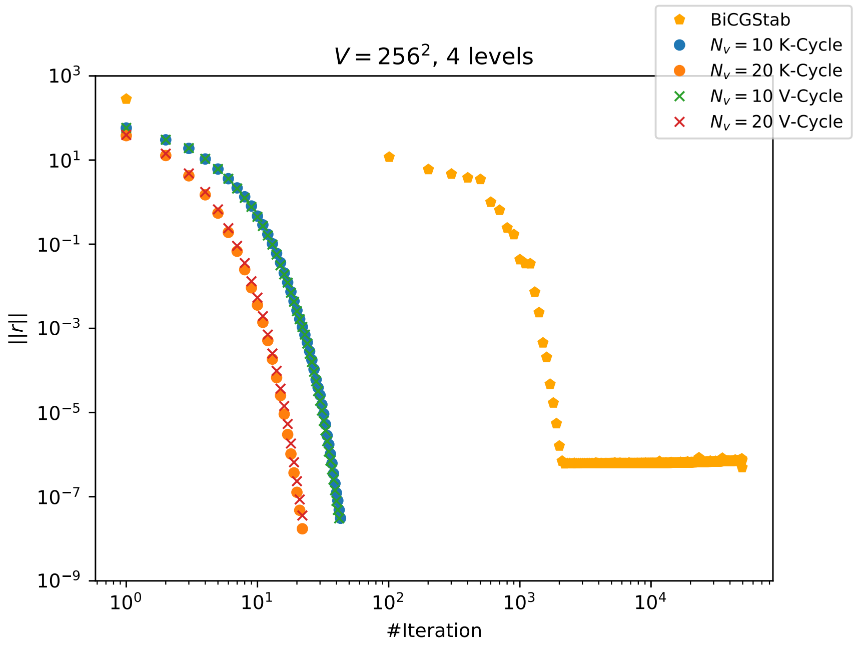
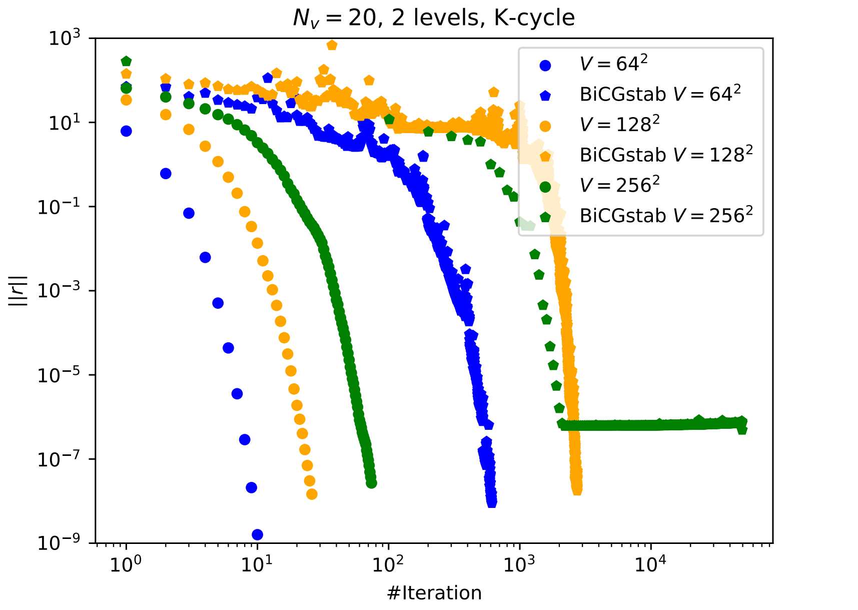
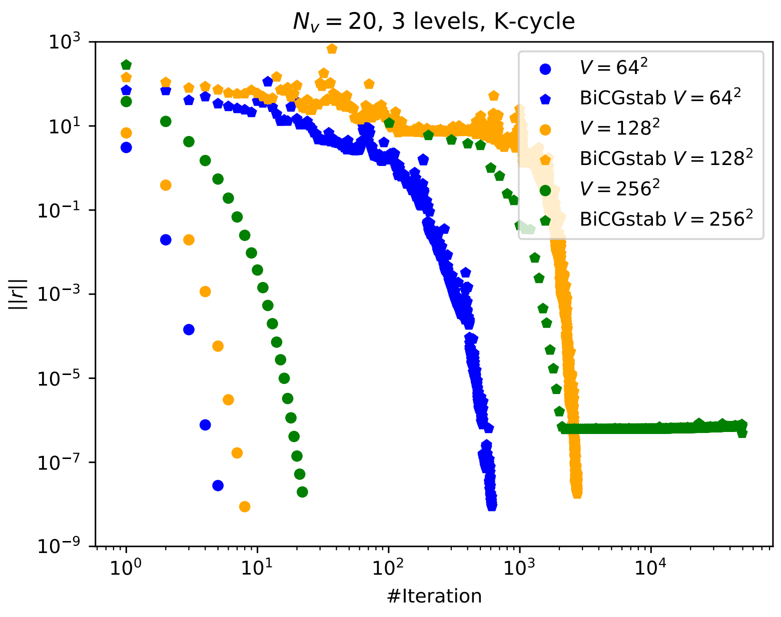


Figure 6: Solution with different levels, $N_v = 10, 20$ and both cycles. We fix the volume to $V = 256^2$.





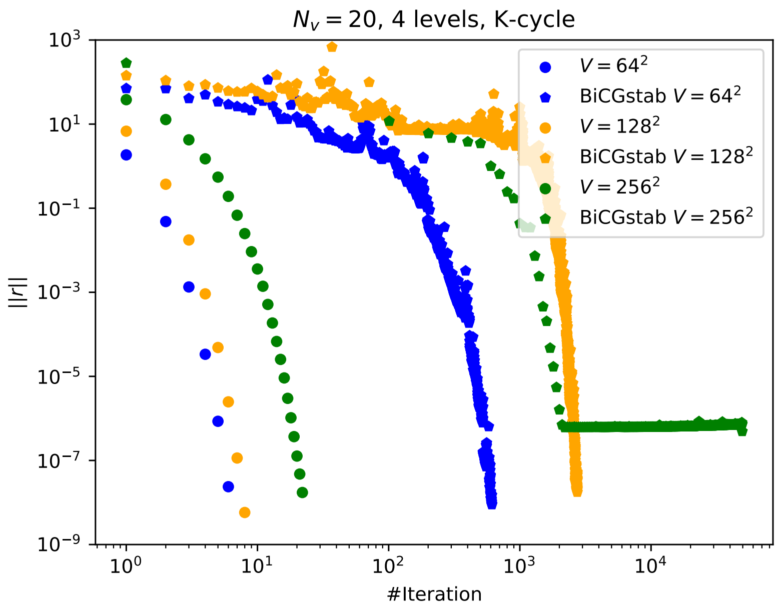
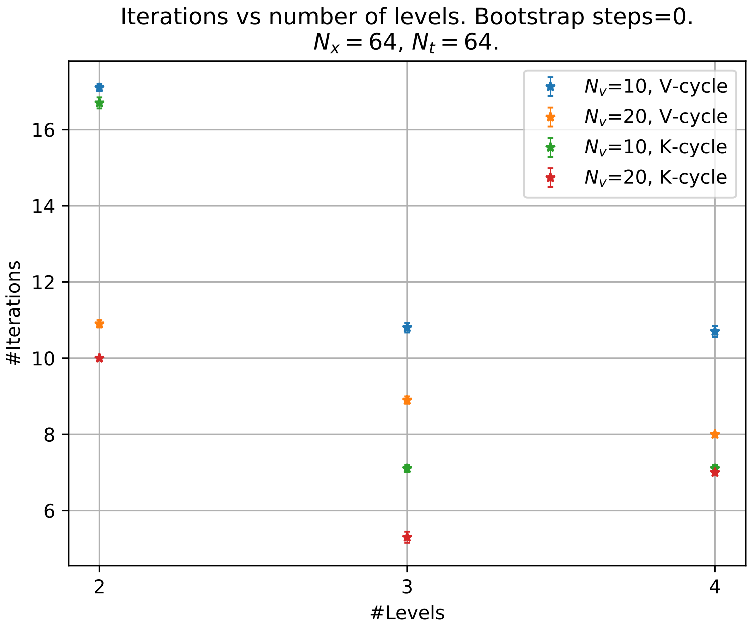
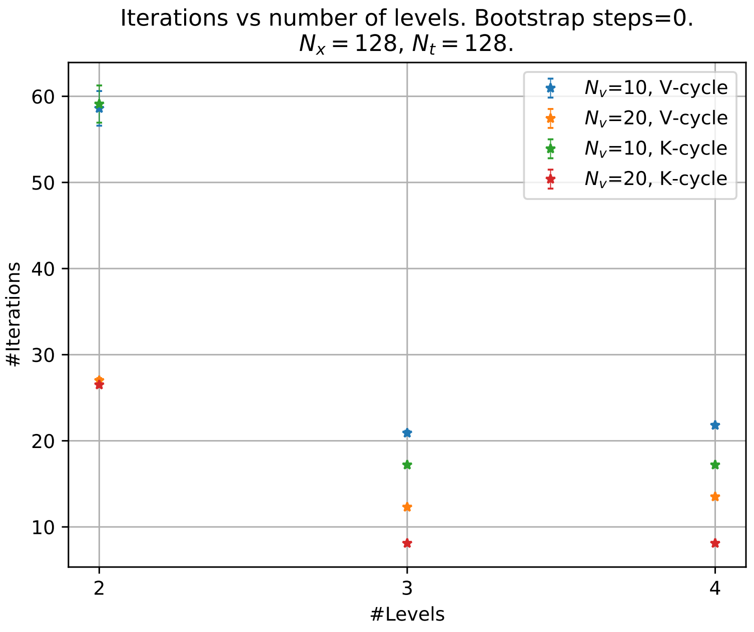


Figure 7: We display the residual for the three volumes using a K-cycle, $N_v = 20$ with 2, 3, and 4 levels.


Iterations vs number of levels. Bootstrap steps=0.
$N_x = 64$, $N_t = 64$.
#Iterations
#Levels
N_v=10, V-cycle
N_v=20, V-cycle
N_v=10, K-cycle
N_v=20, K-cycle


Iterations vs number of levels. Bootstrap steps=0.
$N_x = 128$, $N_t = 128$.
#Iterations
#Levels
N_v=10, V-cycle
N_v=20, V-cycle
N_v=10, K-cycle
N_v=20, K-cycle

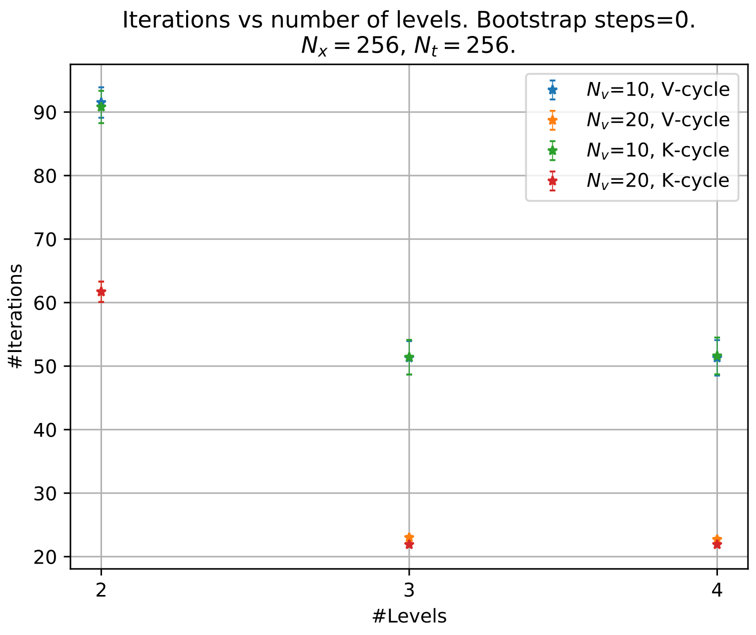
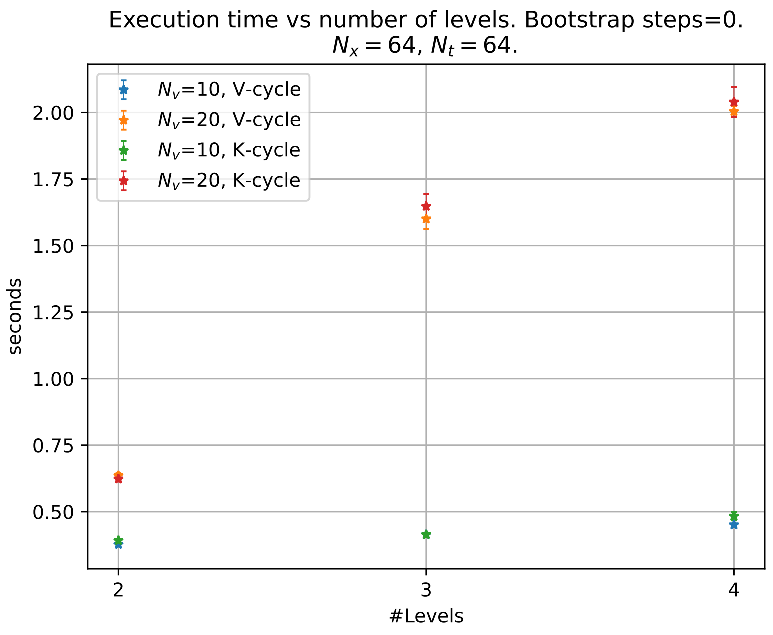
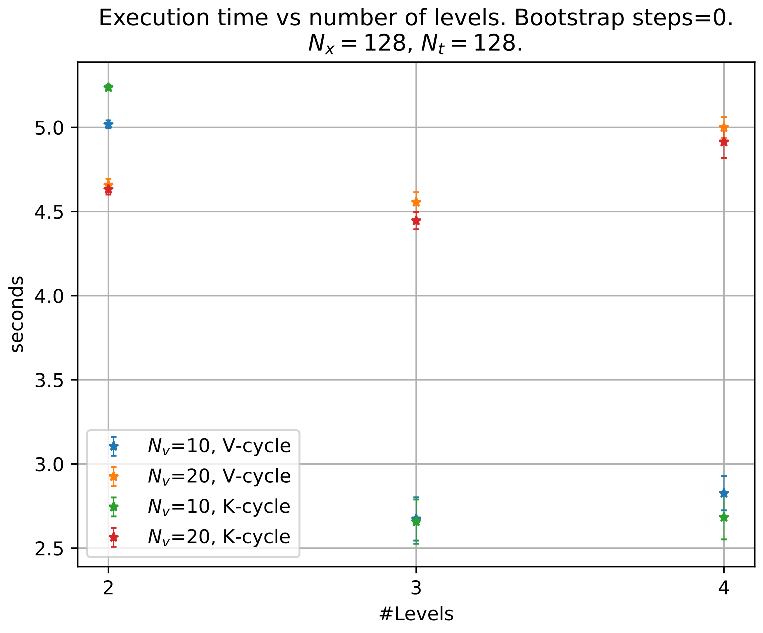


Figure 8: Average number of iterations to achieve a relative tolerance of 10^-10. We average over 10 runs, each one starting with a different set of random test vectors. The gauge configuration and right-hand side are always the same. We apply zero bootstrap steps.





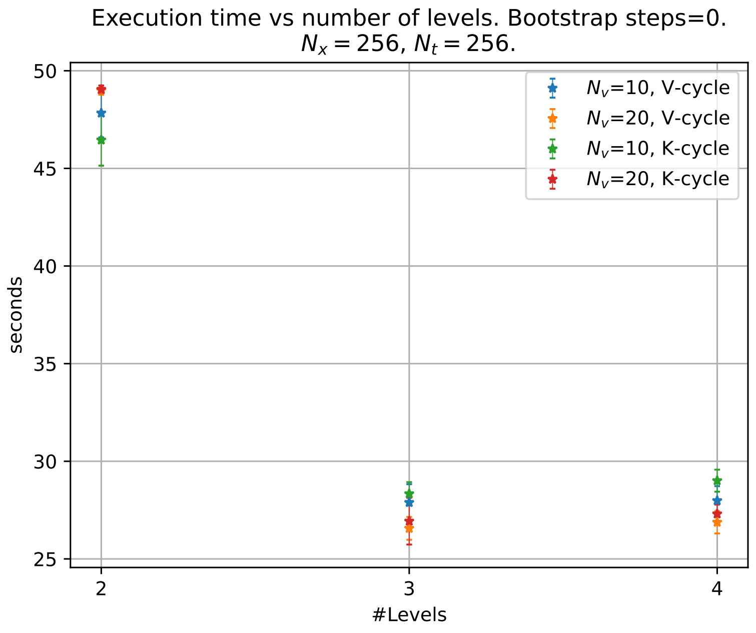
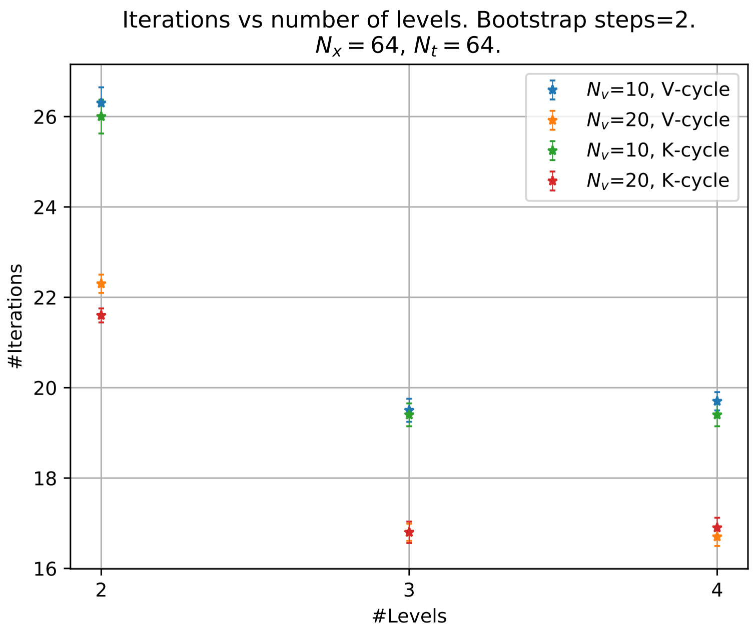
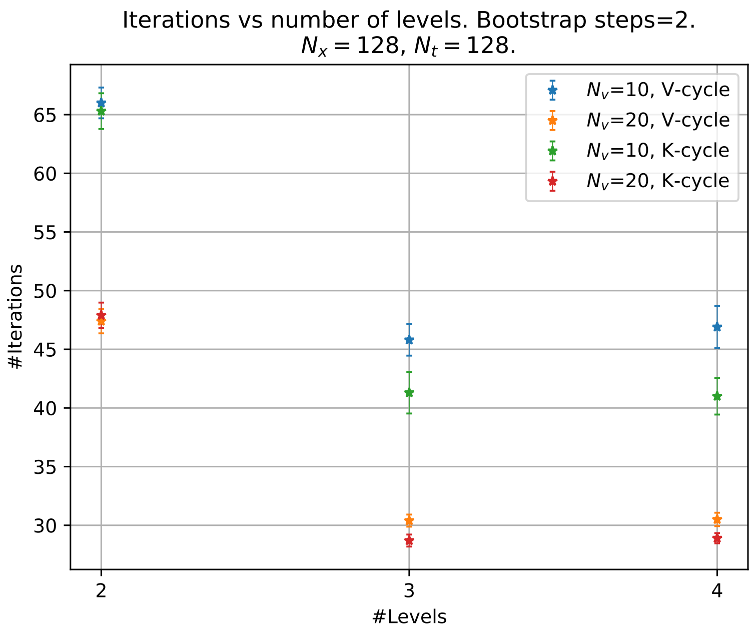


Figure 9: Average execution time to achieve a relative tolerance of 10−10. We average over 10 runs, each one starting with a different set of random test vectors. The gauge configuration and right-hand side are always the same. We apply zero bootstrap steps.


Iterations vs number of levels. Bootstrap steps=2.
$N_x = 64$, $N_t = 64$.
N_v=10, V-cycle
N_v=20, V-cycle
N_v=10, K-cycle
N_v=20, K-cycle
#Iterations
#Levels
16
18
20
22
24
26
2
3
4


Iterations vs number of levels. Bootstrap steps=2.
$N_x = 128$, $N_t = 128$.
N_v=10, V-cycle
N_v=20, V-cycle
N_v=10, K-cycle
N_v=20, K-cycle
#Iterations
#Levels
30
35
40
45
50
55
60
65
2
3
4

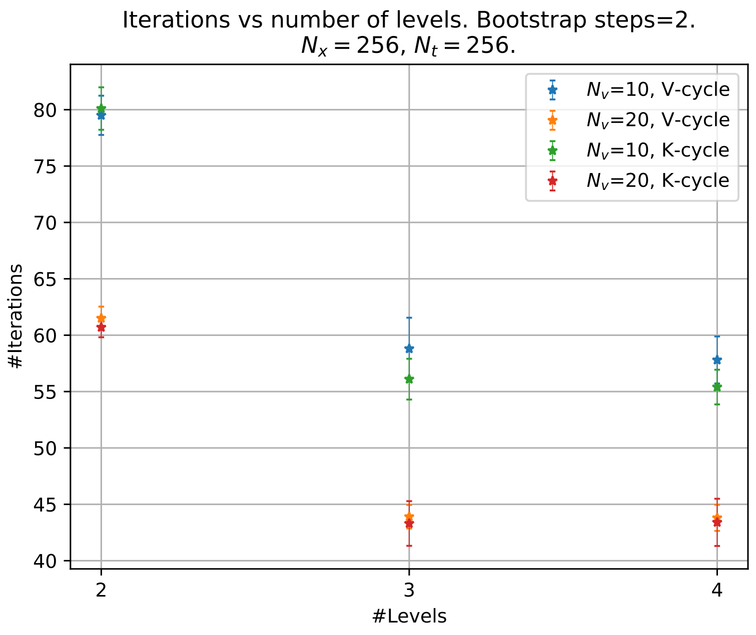
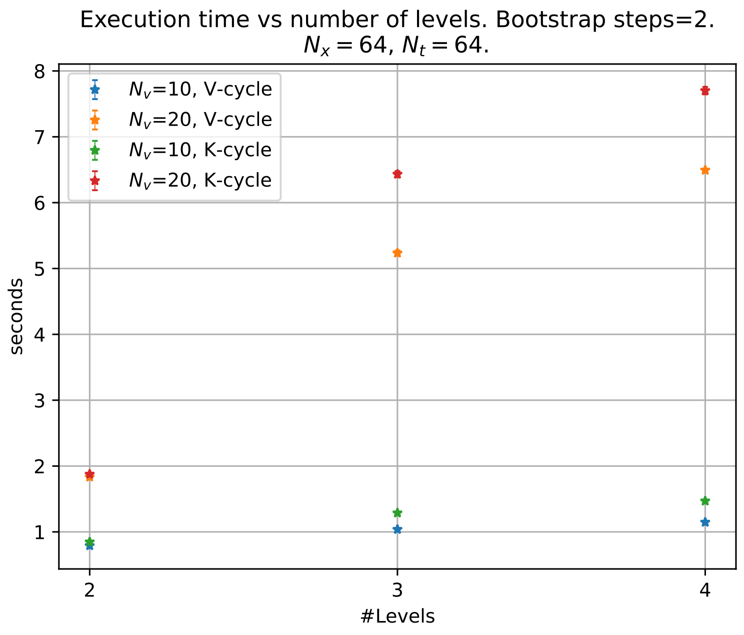
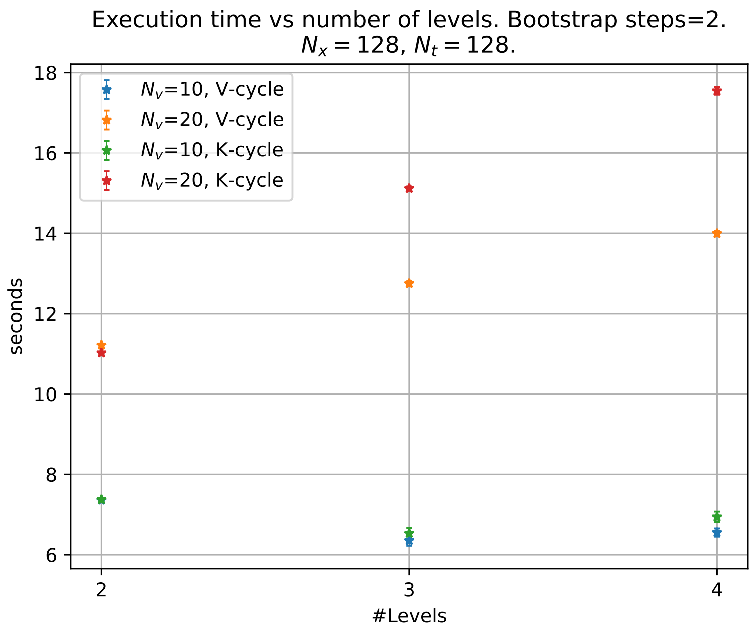


Figure 10: Average number of iterations to achieve a relative tolerance of $10^{-}10$. We average over 10 runs, each one starting with a different set of random test vectors. The gauge configuration and right-hand side are always the same. We apply two bootstrap steps.





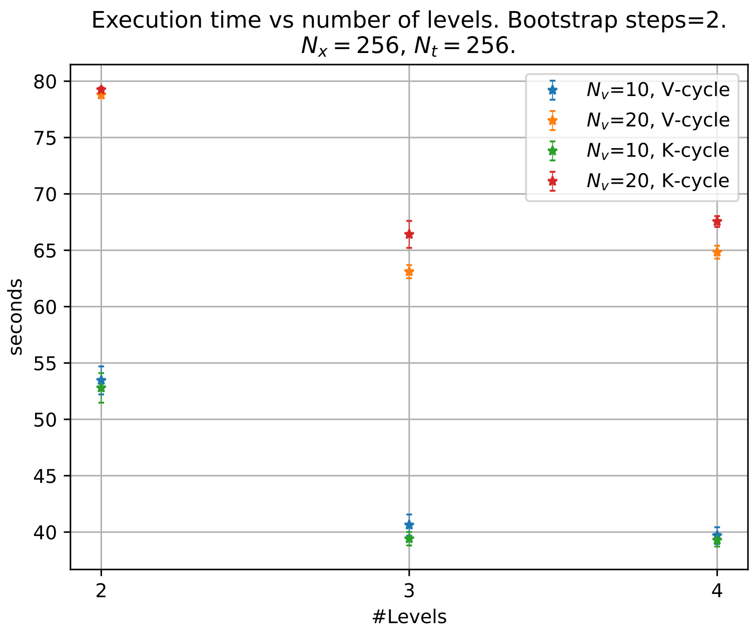


Figure 11: Average execution time to achieve a relative tolerance of 10^-10. We average over 10 runs, each one starting with a different set of random test vectors. The gauge configuration and right-hand side are always the same. We apply two bootstrap steps.

References

[1] N. Christian, K. Jansen, K. Nagai and B. Pollakowski. "Scaling test of the fermion actions in the Schwinger Model", *Nucl. Phys. B*, **739**, (2006).

[2] A. Frommer *et al.* “Adaptive Aggregation-Based Domain Decomposition Multigrid for the Lattice Wilson-Dirac Operator ”, *SIAM*, **36** (2014).

[3] M. Lüscher. “Solution of the Dirac equation in lattice QCD using a domain decomposition method”, *Comput. Phys. Commun.*, **156** (2004).