Note for 2pt calculation with Chroma

Jinchen He

1 Chroma Installation

1.1 Download package

Download necessary packages for Installation from GitHub.

- Use "git clone –recursive . . . ", "recursive" means after the clone is created, initialize all submodules within, using their default settings.
- If the connection to the GitHub is not stable on the server, you are suggested to clone on your local machine, then use "scp" to upload.

Package list:

1. qmp
2. qio
3. qla
4. qdp
5. qopqdp
6. qdpxx
7. chroma

1.2 Configure and make

Configure and make in each folder of packages.

- The whole process can be divided into 7 parts, so that you can locate the errors conveniently.
- "export PATH=. . . :$PATH", makes environment variables available to other programs called from bash.
- "autoreconf -vi": used to update generated configuration files, "-v" means verbosely reporting processing, "-i" means copying missing auxiliary files.
- "./configure", you can use "./configure –help" to see the options
- "./autogen.sh"

1.3 Double v.s. Single

- one place in the "make6.sh" (corresponding to qdpxx)
- two place in the "make7.sh" (corresponding to chroma)
- (two place in the "Makefile" in chroma folder)
- (one place in "chroma-config" in chroma folder)

2 Source code

2.1 Plug in packages

Users are allowed to write some plug in packages and register in the Chroma, so that those packages can be used.

2.2 Make

make sure load corret module (gcc)

- Makefile
- make.sh

2.3 a loop

```
for (int lnum = 0; lnum < params.named_obj.s1_quark_props.size(); ++lnum)
```

This loop is for propagator types, if "s1 quark props" only has one type input, no need to do the loop.

2.4 Point source v.s. Wall source

In the "make source" part of perl script,

1. $< grid > xxxx < /grid >$ means the gaps of source, so wall source should take $< grid > 11196 < /grid >$.
2. $< wvf\_param > x < /wvf\_param >$ means the sigma of gaussian blur, point source needs it.
3. $< wvfIntPar > x < /wvfIntPar >$ means the iteration times of gaussian blur, point source needs it.

Notice: wall source needs gauge fixing (and change to column gauge), but point source doesn't.

2.5 a block

```
for( $id = 0; $id < $n_src; $id = $id + 1 )
{
    if ($id ==0){$it=$it0;}
    else{$it=$it+$t_nsrc;}

print <<"EOF";
    <elem>
      <Name>QPROPADD_cohen</Name>
      <Frequency>1</Frequency>
      <NamedObject>
        <j_decay>3</j_decay>
        <tA>$it  $it</tA>
        <factorA>0.0</factorA>
        <propA>sh_source_dummy</propA>
        <tB>$it  $it</tB>
        <factorB>1.0</factorB>
        <propB>sh_source_ori</propB>
        <propApB>shell_source_$id</propApB>
      </NamedObject>
    </elem>
```

EOF

Explanation: Take parts of propagator and put them together.

2.6 gauge fix

```
  print <<"EOF";

  <elem>
      <!-- Coulomb gauge fix -->
      <Name>COULOMB_GAUGEFIX</Name>
      <Frequency>1</Frequency>
      <Param>
        <version>1</version>
```

```
      <GFAccu>1.0e-6</GFAccu>
      <GFMax>200</GFMax>
      <OrDo>false</OrDo>
      <OrPara>1.0</OrPara>
      <j_decay>3</j_decay>
    </Param>
    <NamedObject>
      <gauge_id>default_gauge_field</gauge_id>
      <gfix_id>column_cfg</gfix_id>
      <gauge_rot_id>gauge_rot</gauge_rot_id>
    </NamedObject>
  </elem>

EOF
$gauge_id="column_cfg";
```

2.7 GPU v.s. CPU

The differences are in the making propagator part of perl script.
For CPU:

```
print <<"EOF";
<elem>
  <Name>PROPAGATOR</Name>
  <Frequency>1</Frequency>
  <Param>
    <version>10</version>
    <quarkSpinType>FULL</quarkSpinType>
    <obsvP>true</obsvP>
    <numRetries>1</numRetries>
      <FermionAction>
        <FermAct>UNPRECONDITIONED_CLOVER</FermAct>
        <Mass>$quark_mass</Mass>
        <clovCoeff>$clover</clovCoeff>
        <FermionBC>
          <FermBC>SIMPLE_FERMBC</FermBC>
          <boundary>1 1 1 -1</boundary>
        </FermionBC>
      </FermionAction>
      <InvertParam>
        <invType>QOP_CLOVER_MULTIGRID_INVERTER</invType>
        <Mass>$quark_mass</Mass>
        <Clover>${clover}</Clover>
        <MaxIter>50</MaxIter>
        <Residual>3e-6</Residual>
        <ExternalSubspace>true</ExternalSubspace>
        <SubspaceId>cpu_multigrid_m$quark_mass</SubspaceId>
        <RsdToleranceFactor>1.5</RsdToleranceFactor>
        <Levels>2</Levels>
        <Blocking>
          <elem>${mg_layout}</elem>
          <elem>2 2 2 1</elem>
        </Blocking>
        <NumNullVecs>24 36</NumNullVecs>
        <NumExtraVecs>0 0</NumExtraVecs>
        <NullResidual>0.4 0.4</NullResidual>
        <NullMaxIter>100 100</NullMaxIter>
        <NullConvergence>0.1 0.1</NullConvergence>
        <Underrelax>1.0 1.0</Underrelax>
        <NumPreHits>0 0</NumPreHits>
        <NumPostHits>4 4</NumPostHits>
```

```
        <CoarseMaxIter>12 12</CoarseMaxIter>
        <CoarseResidual>0.1 0.1</CoarseResidual>
      </InvertParam>
    </Param>
    <NamedObject>
      <gauge_id>$gauge_id</gauge_id>
      <source_id>shell_source_0</source_id>
      <prop_id>prop_m${quark_mass}_p${quark_mom_x}${quark_mom_y}${quark_mom_z}.src0</prop_
    </NamedObject>
  </elem>

EOF
```

For GPU:

```
  print <<"EOF";
  <elem>
  <annotation>
    propagator from the source at ()
  </annotation>
    <Name>PROPAGATOR</Name>
    <Frequency>1</Frequency>
    <Param>
      <version>10</version>
      <quarkSpinType>FULL</quarkSpinType>
      <obsvP>true</obsvP>
      <numRetries>1</numRetries>
        <FermionAction>
          <FermAct>CLOVER</FermAct>
          <Mass>$quark_mass</Mass>
          <clovCoeff>$clover</clovCoeff>
           <FermState>
            <Name>STOUT_FERM_STATE</Name>
            <rho>0.125</rho>
            <n_smear>1</n_smear>
            <orthog_dir>-1</orthog_dir>
            <FermionBC>
              <FermBC>SIMPLE_FERMBC</FermBC>
              <boundary>1 1 1 -1</boundary>
            </FermionBC>
            </FermState>
        </FermionAction>
        <InvertParam>
          <invType>QUDA_MULTIGRID_CLOVER_INVERTER_v2</invType>
          <MULTIGRIDParams>
           <RelaxationOmegaMG>1.0</RelaxationOmegaMG>
           <RelaxationOmegaOuter>1.0</RelaxationOmegaOuter>
           <CheckMultigridSetup>true</CheckMultigridSetup>
            <Residual>1.0e-1</Residual>
            <MaxIterations>12</MaxIterations>
            <MaxCoarseIterations>1000</MaxCoarseIterations>
            <CoarseResidual>0.1</CoarseResidual>
            <Verbosity>true</Verbosity>
            <Precision>SINGLE</Precision>
            <Reconstruct>RECONS_12</Reconstruct>
            <NullVectors>24</NullVectors>
            <GenerateNullspace>true</GenerateNullspace>
            <GenerateAllLevels>true</GenerateAllLevels>
            <CheckMultigridSetup>true</CheckMultigridSetup>
            <CycleType>MG_RECURSIVE</CycleType>
            <Pre-SmootherApplications>0</Pre-SmootherApplications>
            <Post-SmootherApplications>8</Post-SmootherApplications>
```

```
            <SchwarzType>ADDITIVE_SCHWARZ</SchwarzType>
            <Blocking>
              <elem>$mg_layout</elem>
            </Blocking>
          </MULTIGRIDParams>
          <SubspaceID>quda_mg_subspace</SubspaceID>
          <ThresholdCount>500</ThresholdCount>
          <MaxIter>1000</MaxIter>
          <CloverParams>
          <Mass>$quark_mass</Mass>
          <clovCoeff>$clover</clovCoeff>
          <AnisoParam>
            <anisoP>false</anisoP>
            <t_dir>3</t_dir>
            <xi_0>1.0</xi_0>
            <nu>1</nu>
          </AnisoParam>
            <FermionBC>
              <FermBC>SIMPLE_FERMBC</FermBC>
              <boundary>1 1 1 -1</boundary>
            </FermionBC>
          </CloverParams>
          <RsdTarget>$Residual</RsdTarget>
          <Delta>0.1</Delta>
          <RsdToleranceFactor>100</RsdToleranceFactor>
          <AntiPeriodicT>true</AntiPeriodicT>
          <SolverType>CG</SolverType>
          <Verbose>true</Verbose>
          <AsymmetricLinop>true</AsymmetricLinop>
          <CudaReconstruct>RECONS_12</CudaReconstruct>
          <CudaSloppyPrecision>SINGLE</CudaSloppyPrecision>
          <CudaSloppyReconstruct>RECONS_12</CudaSloppyReconstruct>
          <AutotuneDslash>true</AutotuneDslash>
        </InvertParam>
      </Param>
      <NamedObject>
        <gauge_id>$gauge_id</gauge_id>
        <source_id>shell_source_$id</source_id>
        <prop_id>prop_m${quark_mass}_p${quark_mom_x}${quark_mom_y}${quark_mom_z}.src$id</pro
      </NamedObject>
    </elem>

EOF

## after make propagator, erase the useless sources
print <<"EOF";
  <elem>
      <Name>ERASE_NAMED_OBJECT</Name>
      <Frequency>1</Frequency>
      <NamedObject>
          <object_id>shell_source_$id</object_id>
      </NamedObject>
  </elem>

EOF
} # end loop $id

print <<"EOF";
  <elem>
      <Name>ERASE_NAMED_OBJECT</Name>
      <Frequency>1</Frequency>
```

```
    <NamedObject>
        <object_id>sh_source_ori</object_id>
    </NamedObject>
</elem>

<elem>
    <Name>ERASE_NAMED_OBJECT</Name>
    <Frequency>1</Frequency>
    <NamedObject>
        <object_id>sh_source_dummy</object_id>
    </NamedObject>
</elem>

EOF
```

3 Calculation process(in perl script)

1. Parameters input
2. HYP smearing
   do smearing for configuration
3. Make source
4. Make propagator
5. Sink smearing
   do smearing for sink point of the propagator
6. Contracting

4 2pt calculation

4.1 Perl script

Used to print the .xml file as the input for Chroma.
Write perl script as the structure in "xxx.h".

4.1.1 Sink smear

```
print <<"EOF";
<elem>
  <Name>SINK_SMEAR</Name>
  <Frequency>1</Frequency>
  <Param>
    <version>5</version>
    <Sink>
      <version>1</version>
      <SinkType>POINT_SINK</SinkType>
      <j_decay>3</j_decay>
    </Sink>
  </Param>
  <NamedObject>
    <gauge_id>$gauge_id</gauge_id>
    <prop_id>${prop_sum}</prop_id>
    <smeared_prop_id>prop_m${quark_mass}_p${quark_mom_x}${quark_mom_y}
    ${quark_mom_z}.sum_sp</smeared_prop_id>
  </NamedObject>
</elem>
EOF
```

This block read the propagator in $< prop\_id >< /prop\_id >$, then use the method in $< SinkType >< /SinkType >$, and output the smeared propagator as $< smeared\_prop\_id >< /smeared\_prop\_id >$.

4.2 Inline xxx.cc

4.3 Inline xxx.h

In perl script,

```
  <s1_quark_props>
  <elem>prop_m${c_mass}_p000.sum_sp</elem>
</s1_quark_props>
```

So, in the *inline_xxx.h*,

```
multi1d<std::string> s1_quark_props;
```

here just need to read the name of the variable, which is a string, because the *Sinksmear* block in perl script told Chroma to output a smeared propagator with name *xxx.sum_up*, and here we just need to let Chroma know which variable should be used.

4.4 Add new plug in packages

If you want to use a new plug in package in the Chroma for calculation, you should:

1. Write the .cc file and .h file.
2. Put two files above into the source code folder.
3. In the source code folder, add '#include "inline_xxx.h" ' and "foo &= InlinexxxEnv::registerAll();" into "chroma.cc".
4. In the source code folder, add "inline_2pt.h" and "inline_2pt.o" into "Makefile".
5. "bash make.sh" again
6. Update your .pl file to use the new plug in package, and remake the soft link of "chroma" in the same path as .pl file.
7. "sbatch xxx.sh" again.

4.5 Change to use different configurations

If you want to use other different configurations:

1. In the perl script, change "cfg file" and "cfg type", which are configuration path and configuration type.
2. In the perl script, change "ns" and "nt", which are numbers of lattice on space axes and time axis.
3. In the perl script, change "mg layout".
4. In the perl script, change $< tseq > xx < /tseq >$ in the EOF block.
5. In the perl script, change "clover".
6. In the perl and shell, change "quark mass".
7. In the shell, change the list of configurations.

Conf	Size	clovCoeff	mass(140,220,310,670)	mg_layout	geom
A12m310	24×64	1.05088	-0.0785, -0.075,-0.0695,-0.0191	3 3 3 2	1 1 1 4
A12m130	48×64	1.05088	-0.0785, -0.075,-0.0695,-0.0191	4 4 4 4	1 2 3 2
A09m310	32×96	1.04239	-0.058, -0.0554,-0.05138,-0.0174	4 4 4 4	1 1 2 6
A09m130	64×96	1.04239	-0.058, -0.0554,-0.05138,-0.0174	4 4 4 4	2 2 2 6
A06m310	48×144	1.03493	-0.0439,-0.04222,-0.0398, -0.0191	4 4 3 6	1 2 2 6
A06m130	96×192	1.03493	-0.0439,-0.04222,-0.0398, -0.0191	4 4 4 4	3 3 3 12
a045m310	64×192	1.03144	-0.0365(310)	4 4 4 4	2 2 2 12