KAVLI INSTITUTE of Nanoscience Delft

TU Delft

DATA IS NEEDED

QDSIM PROVIDES

A FAST CLASSICAL SIMULATOR FOR QUANTUM DOTS CHARGE STABILITY DIAGRAMS

Valentina Gualtieri, PhD student

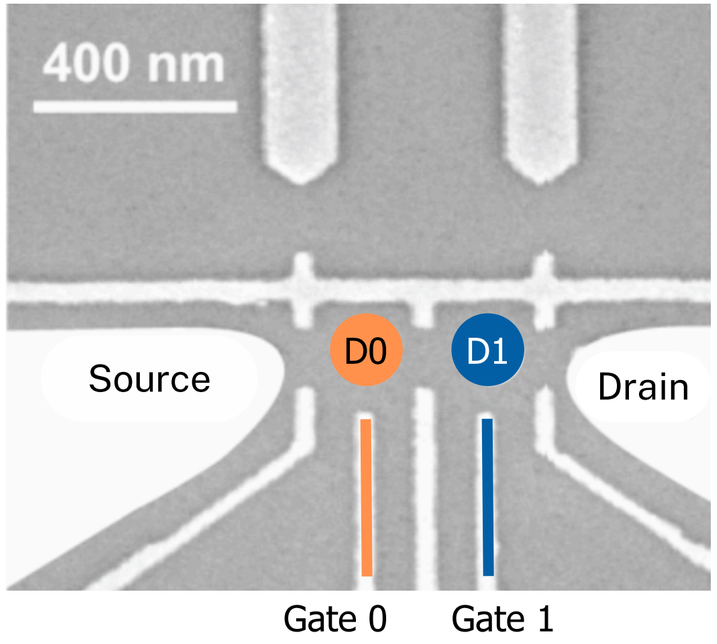
QDsim

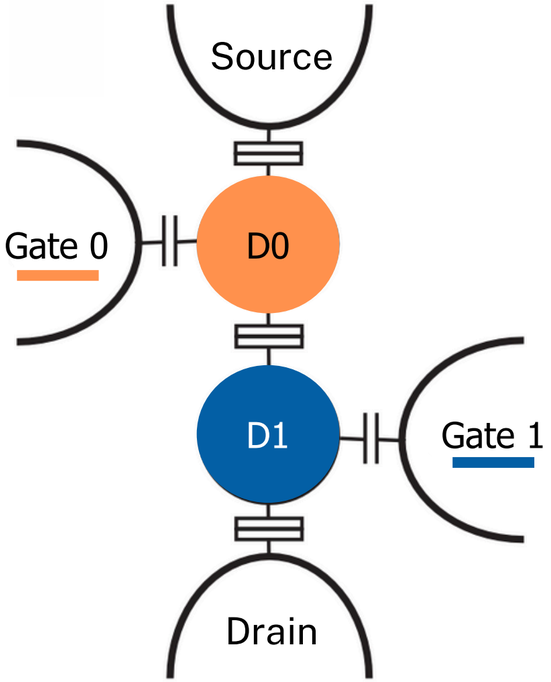
V. Gualtieri, C. Renshaw-Whitman, V. Hernandes, E. Greplova,
arXiv:2404.02712 (2024).

QDsim: Theory

CONSTANT INTERACTION MODEL

L. H. Willems Van Beveren, PhD thesis



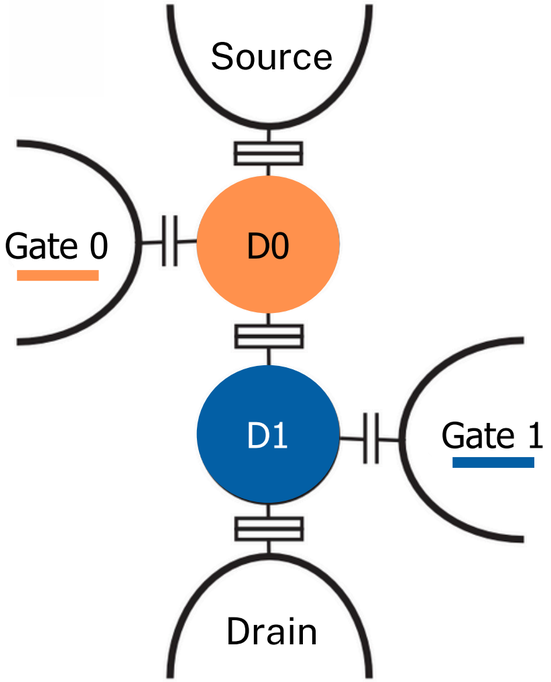


CONSTANT INTERACTION MODEL

$$\begin{bmatrix}\mathbf{Q}_D\\ \mathbf{Q}_G\end{bmatrix} = \begin{bmatrix}\mathbf{C}_{DD} & \mathbf{C}_{DG}\\ \mathbf{C}_{DG}^T & \mathbf{C}_{GG}\end{bmatrix}\begin{bmatrix}\mathbf{V}_D\\ \mathbf{V}_G\end{bmatrix}.$$

$$F = U - W = \frac{1}{2}\left[\mathbf{Q}_D^T, \mathbf{Q}_G^T\right]\begin{bmatrix}\mathbf{V}_D\\ \mathbf{V}_G\end{bmatrix} - \mathbf{V}_G^T\mathbf{Q}_G.$$

$$\begin{aligned}F\left(\mathbf{N}_D;\mathbf{V}_G\right) = &\frac{1}{2}\left(e^2\mathbf{N}_D^T\mathbf{C}_{DD}^{-1}\mathbf{N}_D\right.\\ &-2e\mathbf{N}_D^T\mathbf{C}_{DD}^{-1}\mathbf{C}_{DG}\mathbf{V}_G\\ &\left.+\mathbf{V}_G^T\mathbf{C}_{DG}^T\mathbf{C}_{DD}^{-1}\mathbf{C}_{DG}\mathbf{V}_G\right).\end{aligned}$$



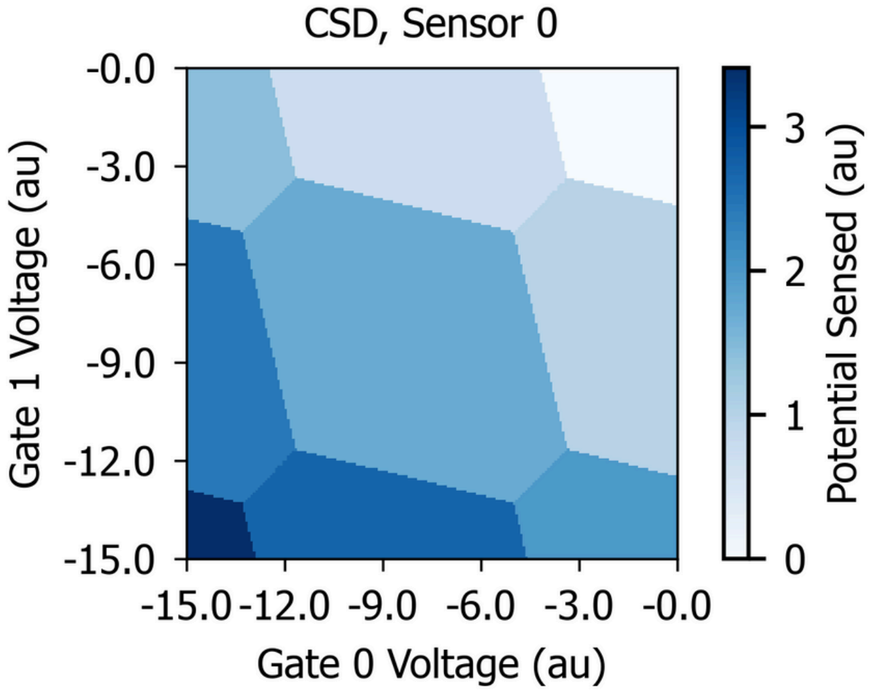
CONSTANT INTERACTION MODEL

$$F_{GS}(\mathbf{V}_G) = \min_{\mathbf{N}\in\mathbb{Z}^{N_D}} F(\mathbf{N};\mathbf{V}_G)$$

$$GS(\mathbf{V}_G) = \{\mathbf{N}\in\mathbb{Z}^{N_D} \mid F(\mathbf{N},\mathbf{V}_G) = F_{GS}(\mathbf{V}_G)\}$$

$$(t^T\mathbf{E}_C\mathbf{C}_{DG})\mathbf{V}_G \preceq \frac{1}{2}t^T\mathbf{E}_C t + t^T\mathbf{E}_C\mathbf{N}_0$$

$$\forall t \in \mathbb{Z}^{N_D}$$

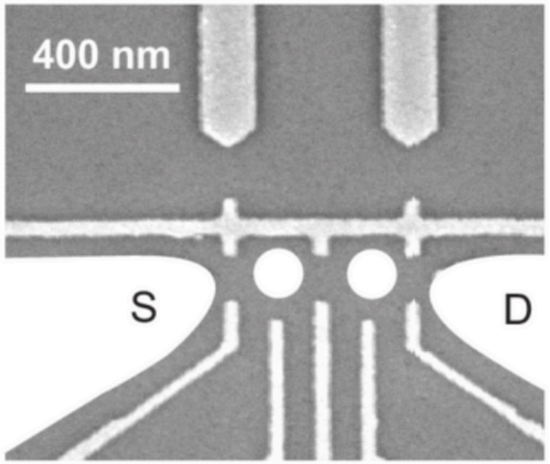


QDsim: Applications

1 **Define the device (*QDDevice* class)**

2 **Define the simulation (*QDSimulator* class)**

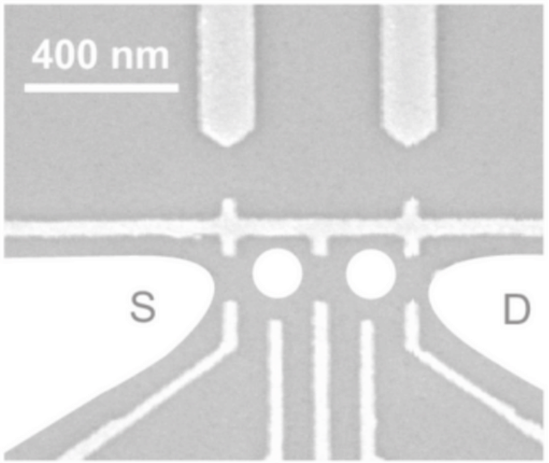
1 Define the device: DOUBLE QD



L. H. Willems Van Beveren, PhD thesis

1 Define the device: DOUBLE QD

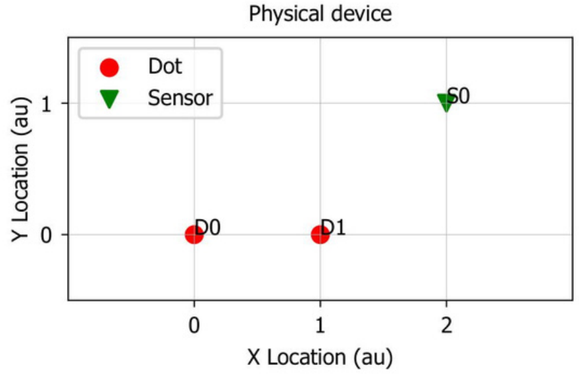
Default



L. H. Willems Van Beveren, PhD thesis

Python Code

```
from qdsim import QDDevice, QDSimulator

# create a quantum dot device object
qddevice = QDDevice()
# double dot
qddevice.one_dimensional_dots_array(
    n_dots=2)
# print the device information
qddevice.print_device_info()
```



Code Output

Device type: in-line array
Number of dots: 2
Number of gates: 2
Physical dot locations: $[(0,0),(1,0)]$
Dot-dot mutual capacitance matrix:

$$\begin{bmatrix} 0.12 & 0.08 \\ 0.08 & 0.12 \end{bmatrix}$$

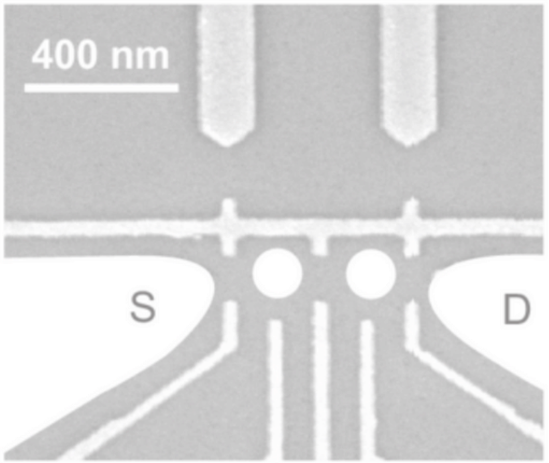
Dot-gate mutual capacitance matrix:

$$\begin{bmatrix} 0.12 & 0.00 \\ 0.00 & 0.12 \end{bmatrix}$$

Reality *Simulation*

1 Define the device: DOUBLE QD

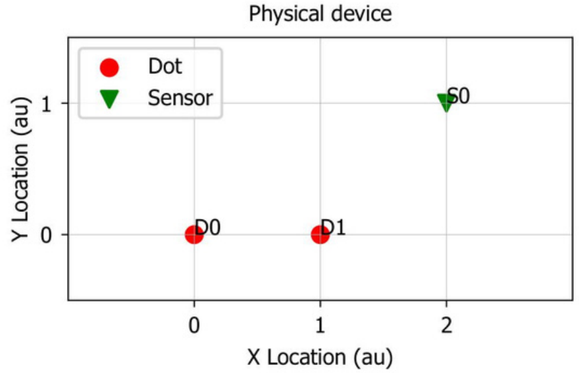
Customization via attributes



L. H. Willems Van Beveren, PhD thesis

Python Code

```
from qdsim import QDDevice, QDSimulator

# create a quantum dot device object
qddevice = QDDevice()
# double dot
qddevice.one_dimensional_dots_array(
    n_dots=2, equal_dots=False,
    equal_gates=False,
    crosstalk_strength=0.3)
# print the device information
qddevice.print_device_info()
```



Code Output

Device type: in-line array
Number of dots: 2
Number of gates: 2
Physical dot locations: $[(0,0),(1,0)]$
Dot-dot mutual capacitance matrix:

$$\begin{bmatrix} 0.12 & 0.08 \\ 0.08 & 0.11 \end{bmatrix}$$

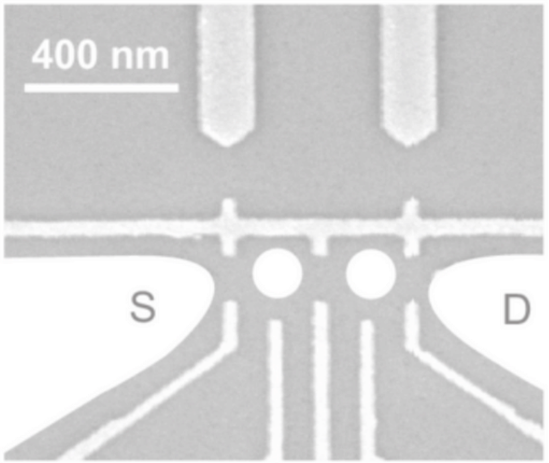
Dot-gate mutual capacitance matrix:

$$\begin{bmatrix} 0.13 & 0.02 \\ 0.02 & 0.15 \end{bmatrix}$$

Reality *Simulation*

1 Define the device: DOUBLE QD

Customization via methods



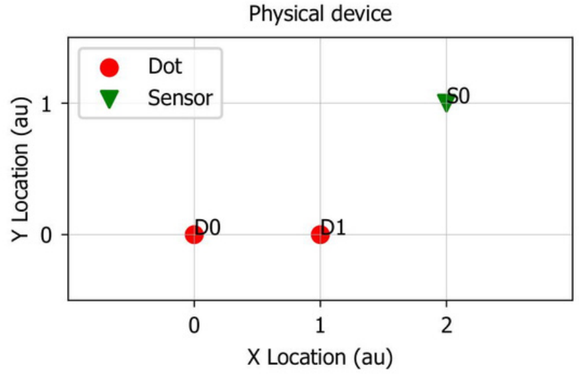
L. H. Willems Van Beveren, PhD thesis

Reality

Python Code

```
from qdsim import QDDevice, QDSimulator

# create a quantum dot device object
qddevice = QDDevice()

# double dot
qddevice.one_dimensional_dots_array(
    n_dots=2)

# define the custom capacitance matrices
cdd = np.array([[0.10, 0.7],[0.7, 0.15]])
cdg = np.array([[0.14, 0.3],[0.3, 0.12]])

# modify the class attributes
qddevice.
    set_dot_dot_mutual_capacitance_matrix(
        cdd)
qddevice.
    set_dot_gate_mutual_capacitance_matrix(
        cdg)

# print the device information
qddevice.print_device_info()
```

Simulation



Code Output

Device type: in-line array
Number of dots: 2
Number of gates: 2
Physical dot locations: $[(0,0),(1,0)]$
Dot-dot mutual capacitance matrix:

$$\begin{bmatrix} 0.10 & 0.07 \\ 0.07 & 0.15 \end{bmatrix}$$

Dot-gate mutual capacitance matrix:

$$\begin{bmatrix} 0.14 & 0.03 \\ 0.03 & 0.12 \end{bmatrix}$$

2 Define the simulation parameters: DOUBLE QD

Python Code

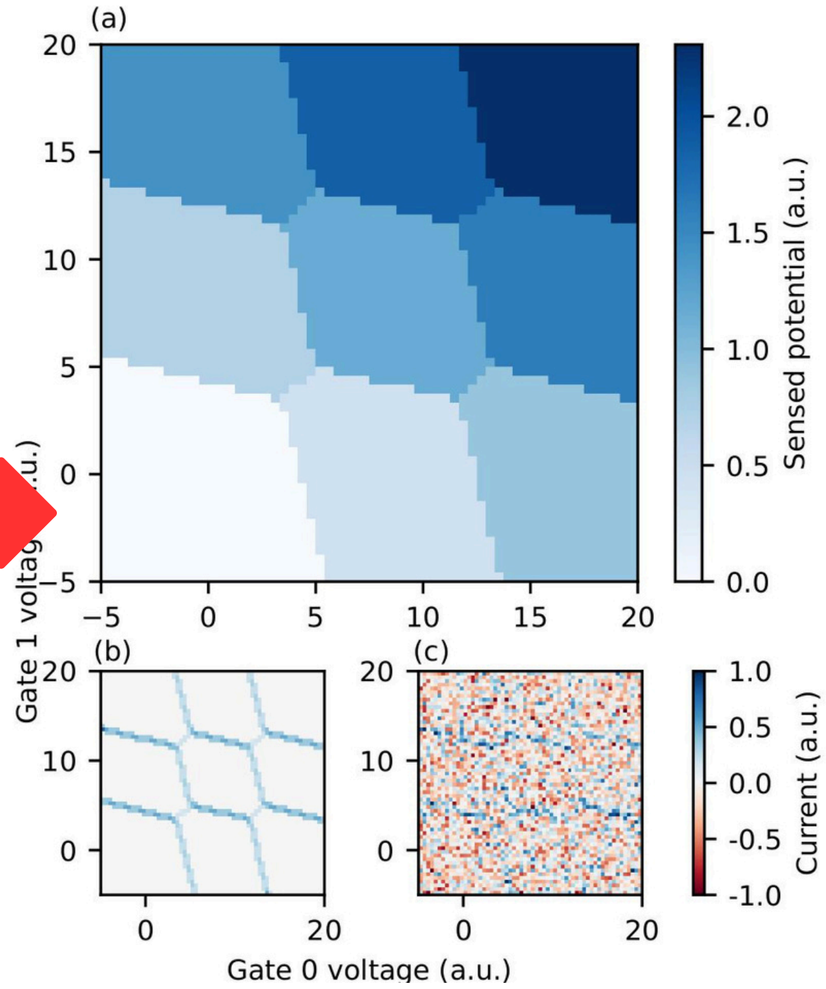
```
# create a quantum dot simulator object
# simulating electrons
qdsimulator = QDSimulator(simulate=
                          'Electrons')

# set the sensor locations
qdsimulator.set_sensor_locations([[2, 1]])

# simulate the charge stability diagram
qdsimulator.
    simulate_charge_stability_diagram(
        qd_device=qddevice, solver='MOSEK',
        v_range_x=[-5, 20],
        v_range_y=[-5, 20],
        n_points_per_axis=60,
        scanning_gate_indexes=[0, 1],
        use_ray=True)
```

Simulation params

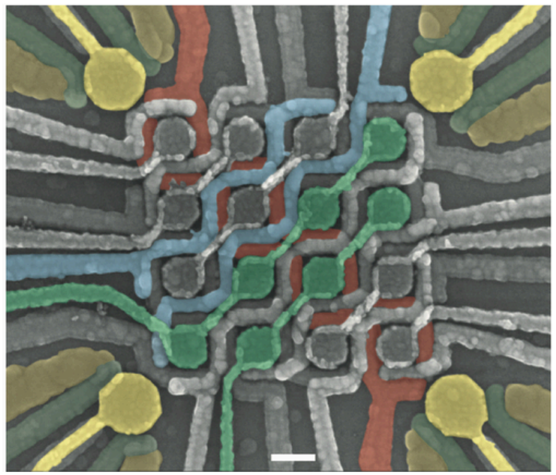
Python Code

```
# plot the charge stability diagram

# potential, no noise
qdsimulator.plot_charge_stability_diagrams(
    cmapvalue='RdBu', plot_potential=True,
    gaussian_noise=False,
    white_noise=False,
    pink_noise=False)

# current, no noise
qdsimulator.plot_charge_stability_diagrams(
    cmapvalue='RdBu', plot_potential=False,
    gaussian_noise=False,
    white_noise=False,
    pink_noise=False)

# current, noisy
qdsimulator.plot_charge_stability_diagrams(
    cmapvalue='RdBu', plot_potential=False,
    gaussian_noise=True,
    white_noise=True,
    pink_noise=True)
```

Plots params



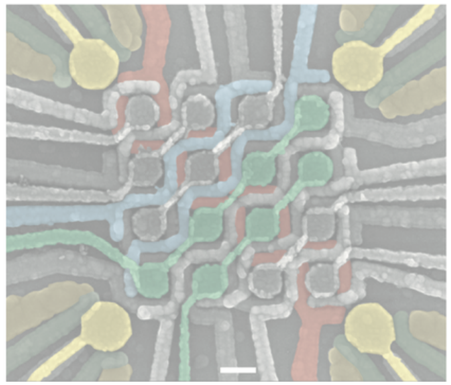
1 Define the device: 4x4 CROSSBAR ARRAY



F. Borsoi et al., Nature Nanotechnology 19, 21 (2024)

1 Define the device: 4x4 CROSSBAR ARRAY

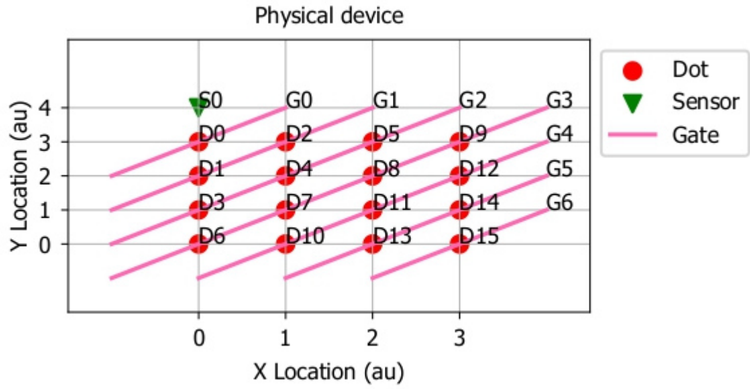
Customization via attributes



F. Borsoi et al., Nature Nanotechnology 19, 21 (2024)

Python Code

```
# create a quantum dot device object
qddevice = QDDevice()

# crossbar array with shared control
# with 4 dots per side
qddevice.crossbar_array_shared_control(
    n_dots_side=4, equal_dots=False,
    equal_gates=False)

# print the device information
qddevice.print_device_info()

# plot device with sensors and save plot
qddevice.plot_device(
    sensor_locations=[[0,4]],
    sensor_labels=['S0'],
    save_plot_to_filepath='4x4_device.pdf')
```



Code Output

Device type: crossbar
Number of dots: 16
Number of gates: 7
Physical dot locations:
[(0, 3), (0, 2), (1, 3), (0, 1), (1, 2), (2, 3), (0, 0), (1, 1), (2, 2), (3, 3), (1, 0), (2, 1), (3, 2), (2, 0), (3, 1), (3, 0)]

Dot-dot mutual capacitance matrix:

$$\begin{bmatrix} 0.12 & 0.08 & 0.08 & 0.00 & 0.04 & 0.00 & 0.00 & 0.00 & 0.00 & 0.00 & 0.00 & 0.00 & 0.00 & 0.00 & 0.00 & 0.00 \\ 0.08 & 0.12 & 0.04 & 0.08 & 0.08 & 0.00 & 0.00 & 0.04 & 0.00 & 0.00 & 0.00 & 0.00 & 0.00 & 0.00 & 0.00 & 0.00 \\ 0.08 & 0.04 & 0.12 & 0.00 & 0.08 & 0.08 & 0.00 & 0.00 & 0.04 & 0.00 & 0.00 & 0.00 & 0.00 & 0.00 & 0.00 & 0.00 \\ 0.00 & 0.08 & 0.00 & 0.12 & 0.04 & 0.00 & 0.08 & 0.08 & 0.00 & 0.00 & 0.04 & 0.00 & 0.00 & 0.00 & 0.00 & 0.00 \\ 0.04 & 0.08 & 0.08 & 0.04 & 0.12 & 0.04 & 0.00 & 0.08 & 0.08 & 0.00 & 0.00 & 0.04 & 0.00 & 0.00 & 0.00 & 0.00 \\ 0.00 & 0.00 & 0.08 & 0.00 & 0.04 & 0.11 & 0.00 & 0.00 & 0.08 & 0.08 & 0.00 & 0.00 & 0.04 & 0.00 & 0.00 & 0.00 \\ 0.00 & 0.00 & 0.00 & 0.08 & 0.00 & 0.00 & 0.12 & 0.04 & 0.00 & 0.00 & 0.08 & 0.00 & 0.00 & 0.00 & 0.00 & 0.00 \\ 0.00 & 0.04 & 0.00 & 0.08 & 0.08 & 0.00 & 0.04 & 0.12 & 0.04 & 0.00 & 0.08 & 0.08 & 0.00 & 0.04 & 0.00 & 0.00 \\ 0.00 & 0.00 & 0.04 & 0.00 & 0.08 & 0.08 & 0.00 & 0.04 & 0.12 & 0.04 & 0.00 & 0.08 & 0.08 & 0.00 & 0.04 & 0.00 \\ 0.00 & 0.00 & 0.00 & 0.04 & 0.00 & 0.00 & 0.08 & 0.00 & 0.04 & 0.12 & 0.00 & 0.00 & 0.08 & 0.00 & 0.00 & 0.00 \\ 0.00 & 0.00 & 0.00 & 0.00 & 0.04 & 0.00 & 0.00 & 0.08 & 0.00 & 0.00 & 0.12 & 0.04 & 0.00 & 0.08 & 0.00 & 0.00 \\ 0.00 & 0.00 & 0.00 & 0.00 & 0.00 & 0.04 & 0.00 & 0.00 & 0.08 & 0.08 & 0.00 & 0.04 & 0.11 & 0.04 & 0.08 & 0.04 \\ 0.00 & 0.00 & 0.00 & 0.00 & 0.00 & 0.04 & 0.00 & 0.00 & 0.08 & 0.08 & 0.00 & 0.04 & 0.11 & 0.00 & 0.08 & 0.00 \\ 0.00 & 0.00 & 0.00 & 0.00 & 0.00 & 0.00 & 0.04 & 0.00 & 0.00 & 0.08 & 0.08 & 0.00 & 0.00 & 0.11 & 0.04 & 0.08 \\ 0.00 & 0.00 & 0.00 & 0.00 & 0.00 & 0.00 & 0.00 & 0.04 & 0.00 & 0.00 & 0.08 & 0.08 & 0.04 & 0.12 & 0.08 & 0.00 \\ 0.00 & 0.00 & 0.00 & 0.00 & 0.00 & 0.00 & 0.00 & 0.00 & 0.00 & 0.00 & 0.04 & 0.00 & 0.08 & 0.08 & 0.12 & 0.00 \end{bmatrix}$$

Dot-gate mutual capacitance matrix:

$$\begin{bmatrix} 0.15 & 0.00 & 0.00 & 0.00 & 0.00 & 0.00 & 0.00 \\ 0.00 & 0.18 & 0.00 & 0.00 & 0.00 & 0.00 & 0.00 \\ 0.00 & 0.18 & 0.00 & 0.00 & 0.00 & 0.00 & 0.00 \\ 0.00 & 0.00 & 0.12 & 0.00 & 0.00 & 0.00 & 0.00 \\ 0.00 & 0.00 & 0.12 & 0.00 & 0.00 & 0.00 & 0.00 \\ 0.00 & 0.00 & 0.12 & 0.00 & 0.00 & 0.00 & 0.00 \\ 0.00 & 0.00 & 0.00 & 0.13 & 0.00 & 0.00 & 0.00 \\ 0.00 & 0.00 & 0.00 & 0.13 & 0.00 & 0.00 & 0.00 \\ 0.00 & 0.00 & 0.00 & 0.13 & 0.00 & 0.00 & 0.00 \\ 0.00 & 0.00 & 0.00 & 0.13 & 0.00 & 0.00 & 0.00 \\ 0.00 & 0.00 & 0.00 & 0.00 & 0.13 & 0.00 & 0.00 \\ 0.00 & 0.00 & 0.00 & 0.00 & 0.13 & 0.00 & 0.00 \\ 0.00 & 0.00 & 0.00 & 0.00 & 0.13 & 0.00 & 0.00 \\ 0.00 & 0.00 & 0.00 & 0.00 & 0.00 & 0.16 & 0.00 \\ 0.00 & 0.00 & 0.00 & 0.00 & 0.00 & 0.16 & 0.00 \\ 0.00 & 0.00 & 0.00 & 0.00 & 0.00 & 0.00 & 0.13 \end{bmatrix}$$

Reality

Simulation

2 Define the simulation parameters: DOUBLE QD

Python Code

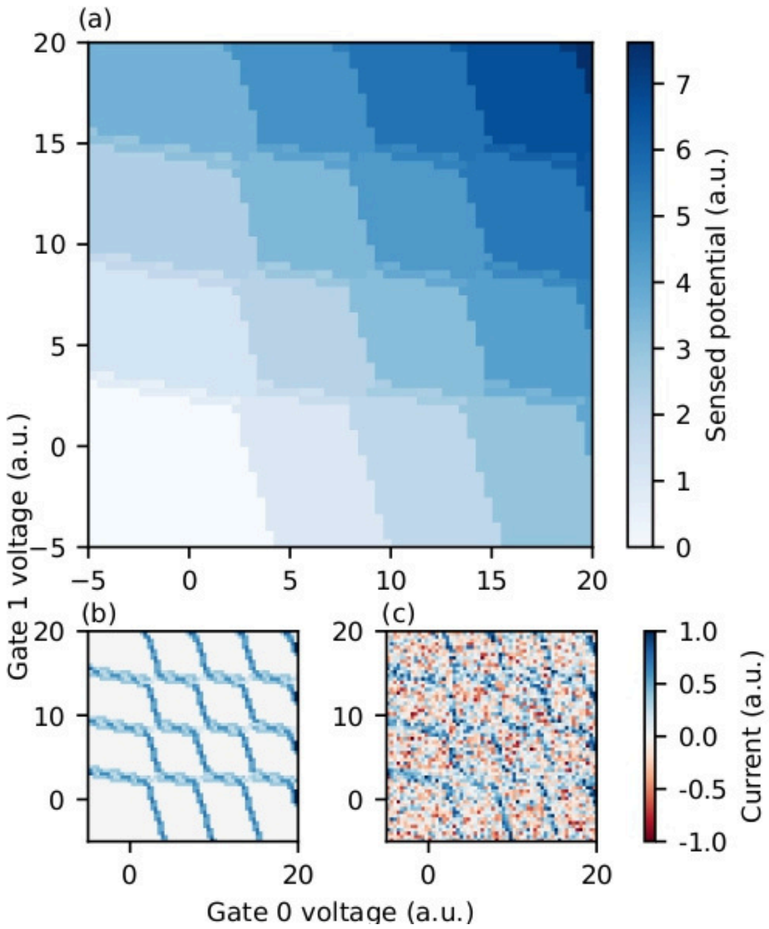
```
# create a quantum dot simulator object
# simulating electrons
qdsimulator = QDSimulator(simulate=
                          'Electrons')

# set the same sensor locations
qdsimulator.set_sensor_locations([[0, 4]])

# simulate the charge stability diagram
qdsimulator.
    simulate_charge_stability_diagram(
        qd_device=qddevice, solver='MOSEK',
        v_range_x=[-5, 20],
        v_range_y=[-5, 20],
        n_points_per_axis=60,
        scanning_gate_indexes=[0, 1],
        fixed_voltage=1, use_ray=True)
```

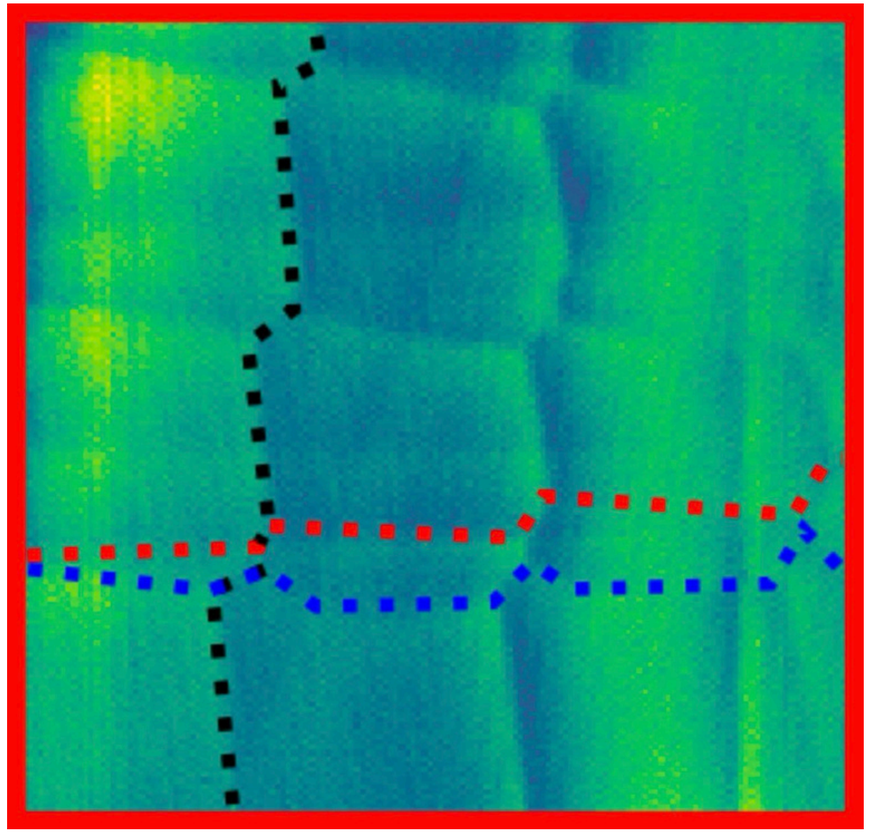
Simulation params

Python Code

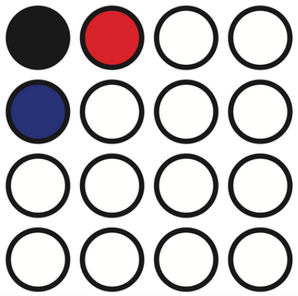
```
# plot the charge stability diagram

# potential, no noise
qdsimulator.plot_charge_stability_diagrams(
    cmapvalue='RdBu', plot_potential=True,
    gaussian_noise=False,
    white_noise=False,
    pink_noise=False)

# current, no noise
qdsimulator.plot_charge_stability_diagrams(
    cmapvalue='RdBu', plot_potential=False,
    gaussian_noise=False,
    white_noise=False,
    pink_noise=False)

# current, noisy
qdsimulator.plot_charge_stability_diagrams(
    cmapvalue='RdBu', plot_potential=False,
    gaussian_noise=True,
    white_noise=True,
    pink_noise=True)
```

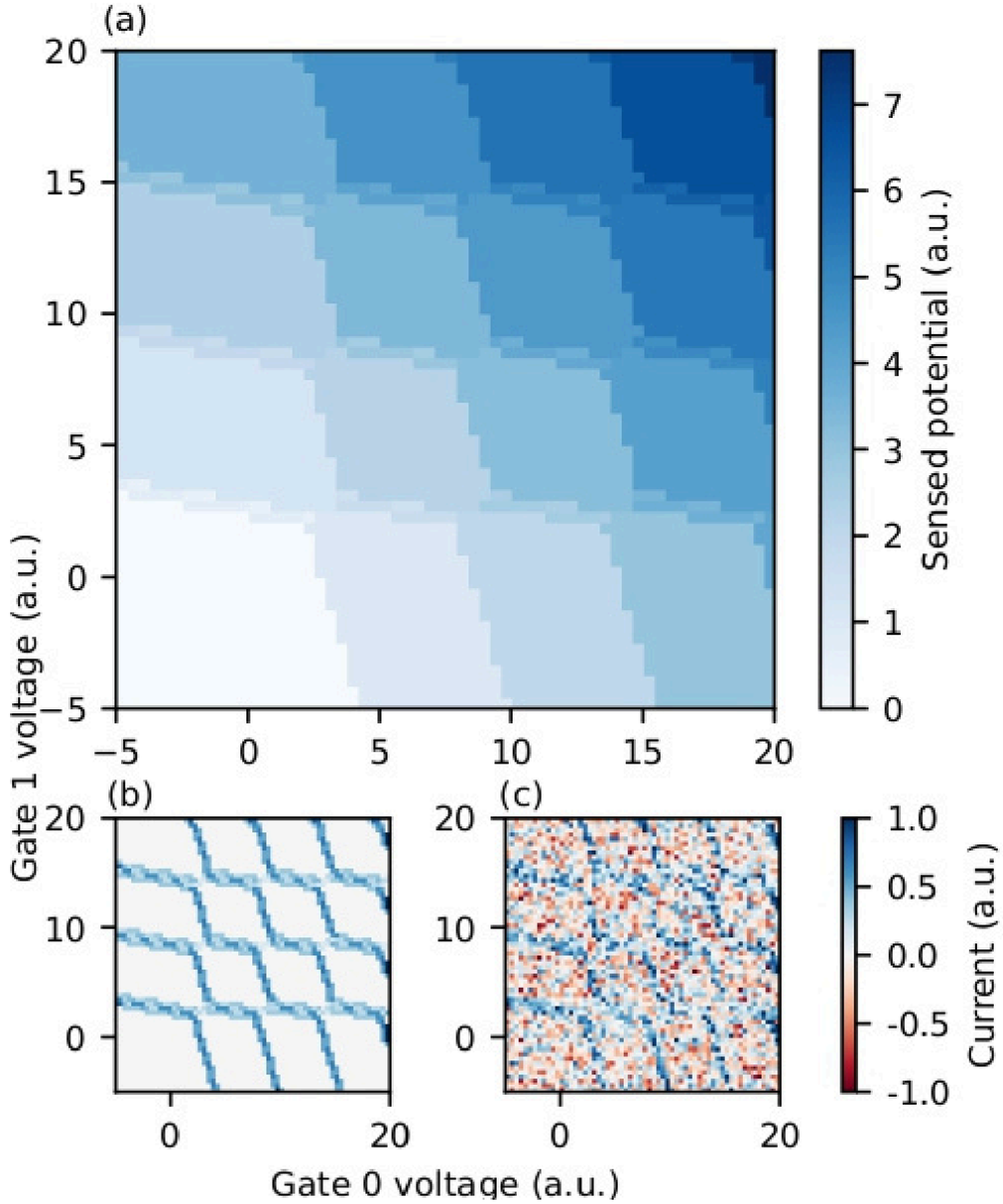
Plots params





F. Borsoi et al., Nature Nanotechnology 19, 21 (2024)





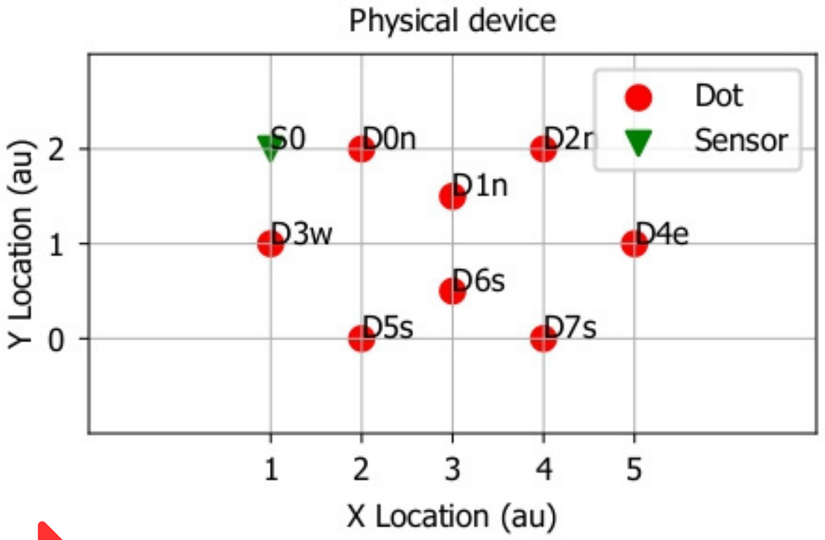
1 Define the device: CUSTOM DESIGN

Python Code

```
# create a quantum dot device object
qddevice = QDDevice()

# set the custom dot locations
qddevice.set_custom_dot_locations([[2, 2],
    [3, 1.5], [4, 2], [1, 1], [5, 1],
    [2, 0], [3, 0.5], [4, 0]],
    equal_dots=False, equal_gates=False,
    crosstalk_strength=0.2, c0=0.12)

# plot the device with custom labels
# and sensor
qddevice.plot_device(
    sensor_locations=[[1,2]],
    sensor_labels=['S0'],
    custom_dot_labels=['D0n', 'D1n', 'D2n',
        'D3w','D4e', 'D5s','D6s', 'D7s'])
```

Python Code

```
dot_gate_matrix = np.array([
    [0.14, 0.00, 0.00, 0.00],
    [0.14, 0.00, 0.00, 0.00],
    [0.14, 0.00, 0.00, 0.00],
    [0.00, 0.13, 0.00, 0.00],
    [0.00, 0.00, 0.12, 0.00],
    [0.00, 0.00, 0.00, 0.15],
    [0.00, 0.00, 0.00, 0.15],
    [0.00, 0.00, 0.00, 0.15]
])

qddevice.
    set_dot_gate_mutual_capacitance_matrix(
        dot_gate_matrix)

qddevice.print_device_info()
```



Code Output

Device type: custom
Number of dots: 8
Number of gates: 4
Physical dot locations:
$[[2, 2], [3, 1.5], [4, 2], [1, 1], [5, 1], [2, 0], [3, 0.5], [4, 0]]$
Dot-dot mutual capacitance matrix:

$$\begin{bmatrix} 0.12 & 0.06 & 0.00 & 0.04 & 0.00 & 0.00 & 0.00 & 0.00 \\ 0.06 & 0.12 & 0.06 & 0.00 & 0.00 & 0.00 & 0.08 & 0.00 \\ 0.00 & 0.06 & 0.12 & 0.00 & 0.04 & 0.00 & 0.00 & 0.00 \\ 0.04 & 0.00 & 0.00 & 0.12 & 0.00 & 0.04 & 0.00 & 0.00 \\ 0.00 & 0.00 & 0.04 & 0.00 & 0.12 & 0.00 & 0.00 & 0.04 \\ 0.00 & 0.00 & 0.00 & 0.04 & 0.00 & 0.12 & 0.06 & 0.00 \\ 0.00 & 0.08 & 0.00 & 0.00 & 0.00 & 0.06 & 0.12 & 0.06 \\ 0.00 & 0.00 & 0.00 & 0.00 & 0.04 & 0.00 & 0.06 & 0.12 \end{bmatrix}$$

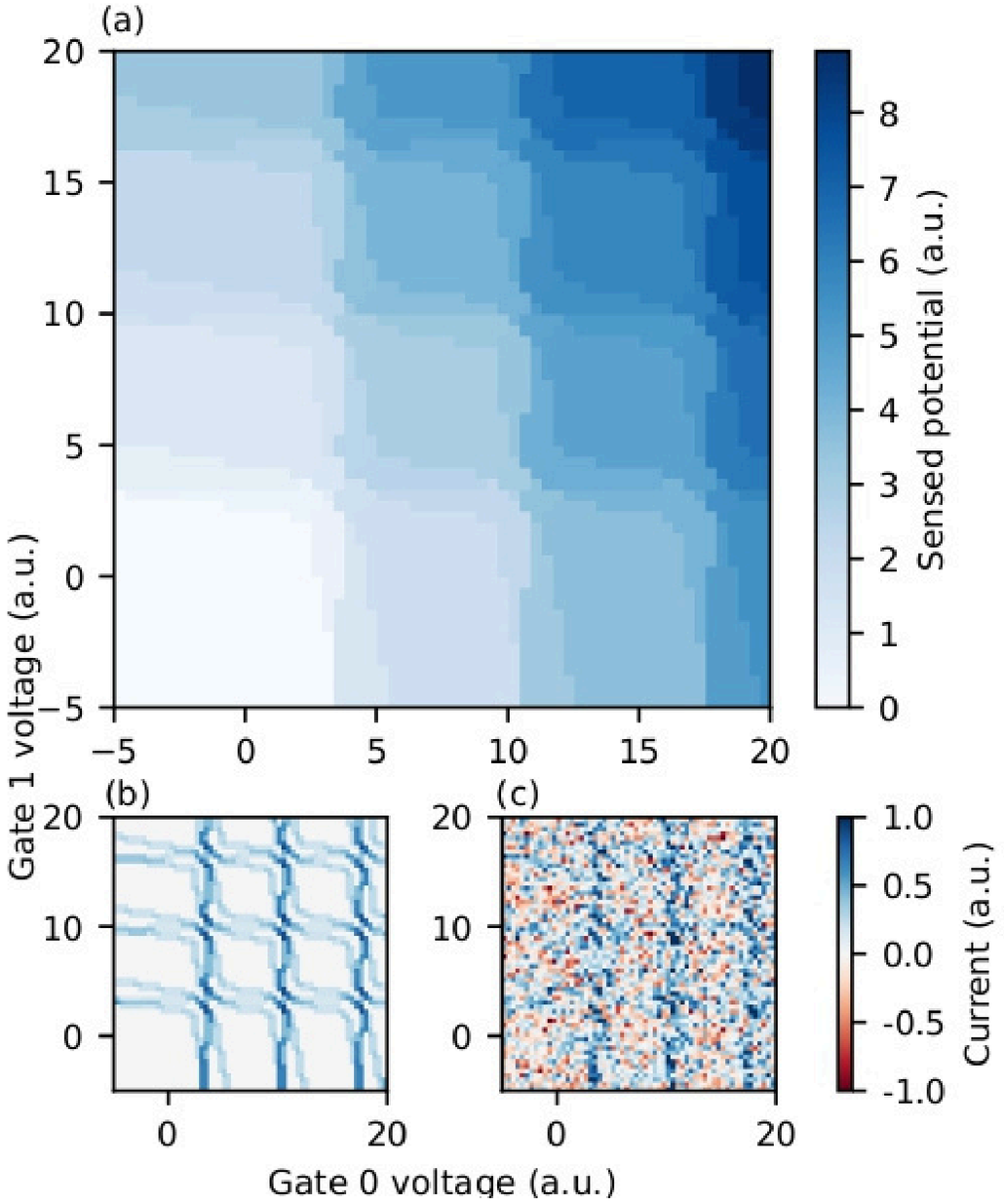
Dot-gate mutual capacitance matrix:

$$\begin{bmatrix} 0.14 & 0.00 & 0.00 & 0.00 \\ 0.14 & 0.00 & 0.00 & 0.00 \\ 0.14 & 0.00 & 0.00 & 0.00 \\ 0.00 & 0.13 & 0.00 & 0.00 \\ 0.00 & 0.00 & 0.12 & 0.00 \\ 0.00 & 0.00 & 0.00 & 0.15 \\ 0.00 & 0.00 & 0.00 & 0.15 \\ 0.00 & 0.00 & 0.00 & 0.15 \end{bmatrix}$$

2 Define the simulation parameters: DOUBLE QD

Python Code

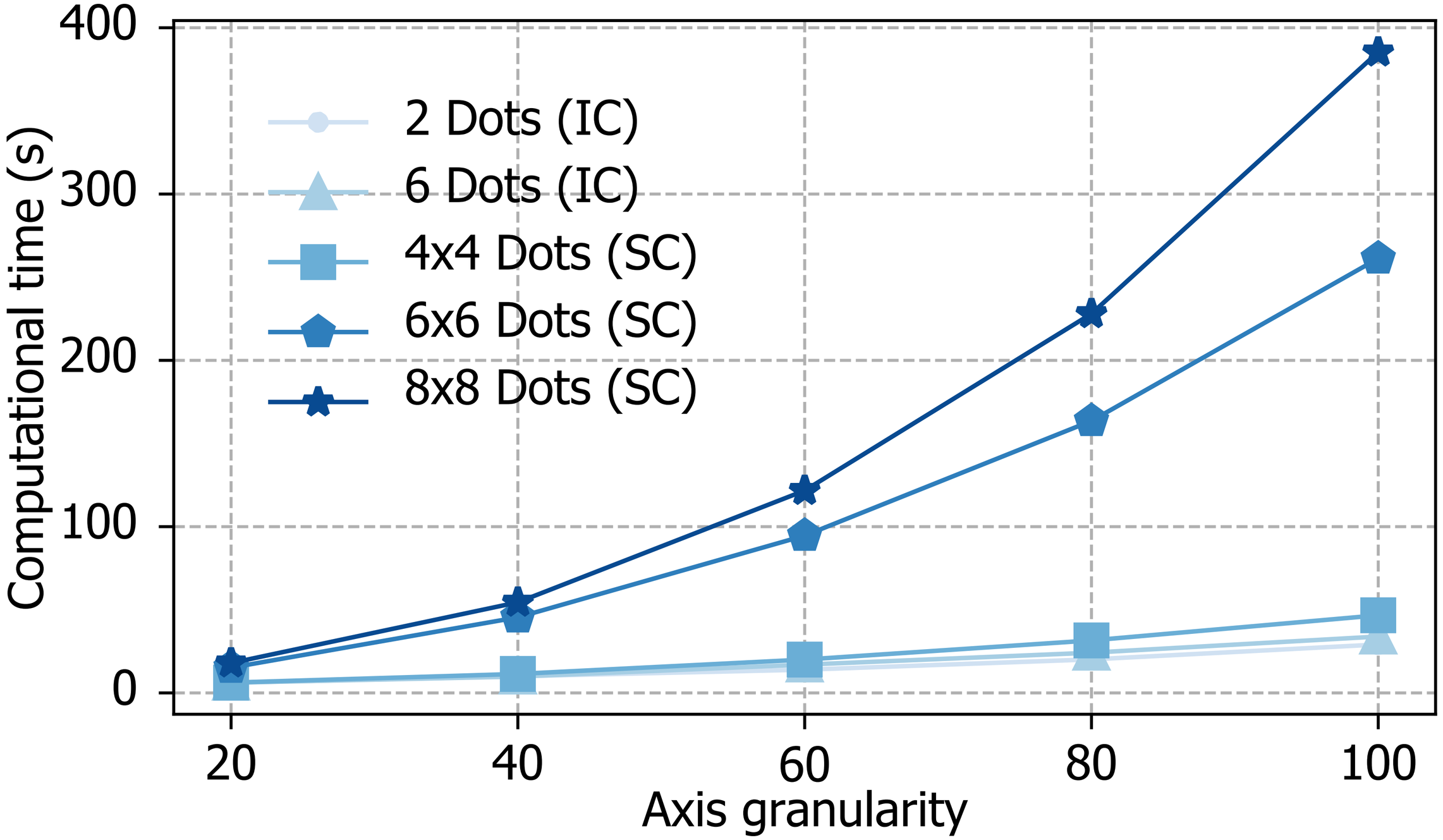
```
# create a quantum dot simulator object
qdsimulator = QDSimulator(simulate=
                          'Electrons')

# set the sensor locations
qdsimulator.set_sensor_locations([[2, 1]])

# simulate the charge stability diagram
qdsimulator.
    simulate_charge_stability_diagram(
        qd_device=qddevice, solver='MOSEK',
        v_range_x=[-5, 20],
        v_range_y=[-5, 20],
        n_points_per_axis=60,
        scanning_gate_indexes=[0, 3],
        use_ray=True)

# plot the charge stability diagrams
qdsimulator.plot_charge_stability_diagrams(
    cmapvalue='RdBu',plot_potential=True,
    gaussian_noise=False, white_noise=False,
    pink_noise=False)

qdsimulator.plot_charge_stability_diagrams(
    cmapvalue='RdBu',plot_potential=False,
    gaussian_noise=False, white_noise=False,
    pink_noise=False)

qdsimulator.plot_charge_stability_diagrams(
    cmapvalue='RdBu', plot_potential=False,
    gaussian_noise=True, white_noise=True,
    pink_noise=True)
```



Characteristics:

DOES NOT account for quantum mechanical effects

VERY FAST
Can simulate large arrays


400
300
200
100
0
Computational time (s)
2 Dots (IC)
6 Dots (IC)
4x4 Dots (SC)
6x6 Dots (SC)
8x8 Dots (SC)
20
40
60
80
100
Axis granularity

Thank You