Supplementary Information for:

A high-throughput reactor array applied to the parameter exploration of copper-exchanged zeolites for dilute methane oxidation

Martin, Elijah E.[1], Sawyer, William J.[1,2], Westendorff, Karl S. [3], Hart, A. John[2], Plata, Desirée L.[1]

[1]Department of Civil and Environmental Engineering,

[2]Department of Mechanical Engineering,

[3]Department of Chemical Engineering, Massachusetts Institute of Technology, Cambridge, MA, USA, 02139

Table S1

Web of science search words	Total publications	Publications in 2004	Publicationss in 2024
"high-throughput experimentation" (All Fields)	1447	30	159
"high-throughput" (All Fields) and "reactor" (All Fields) and "catalysis" (All Fields)	317	16	18

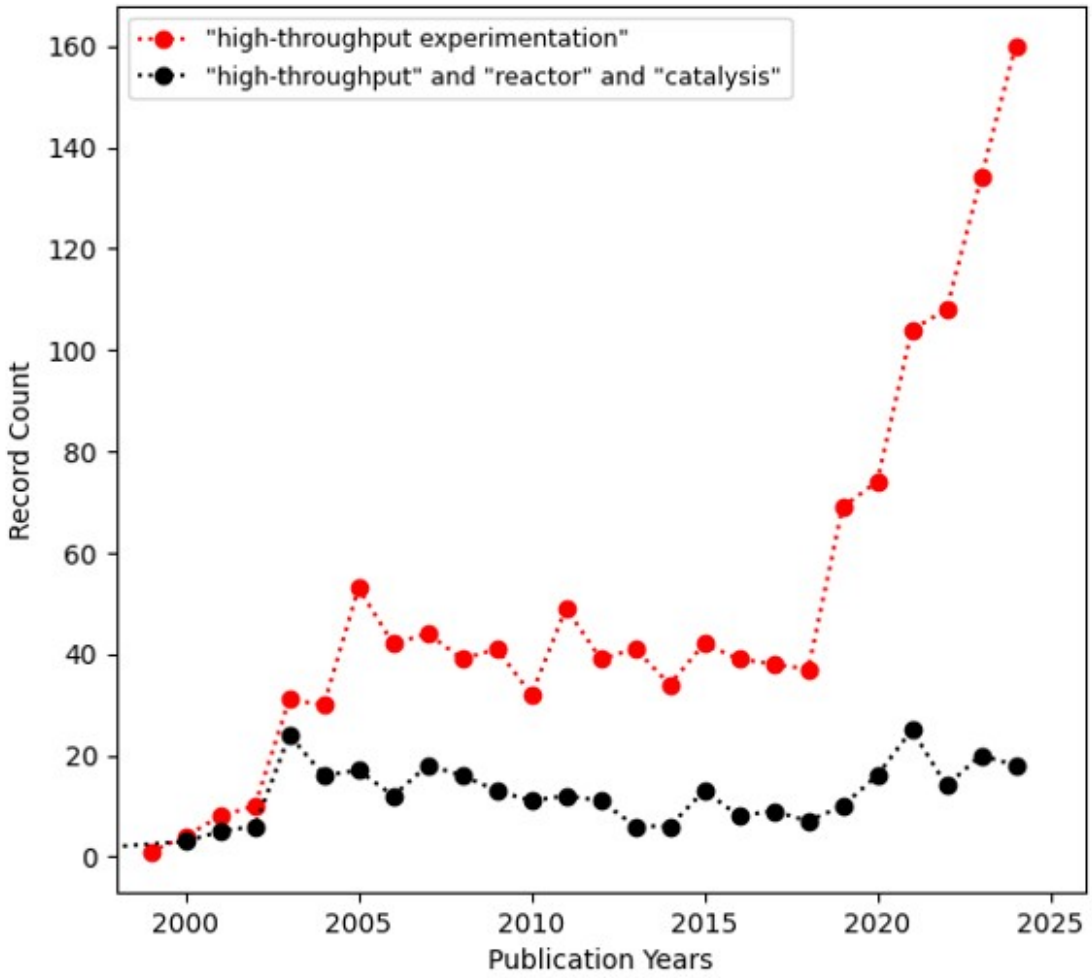


Figure S1. Web of Science publication rate history of two searches. Search descriptors are displayed in the legend and are used to query all searchable fields.

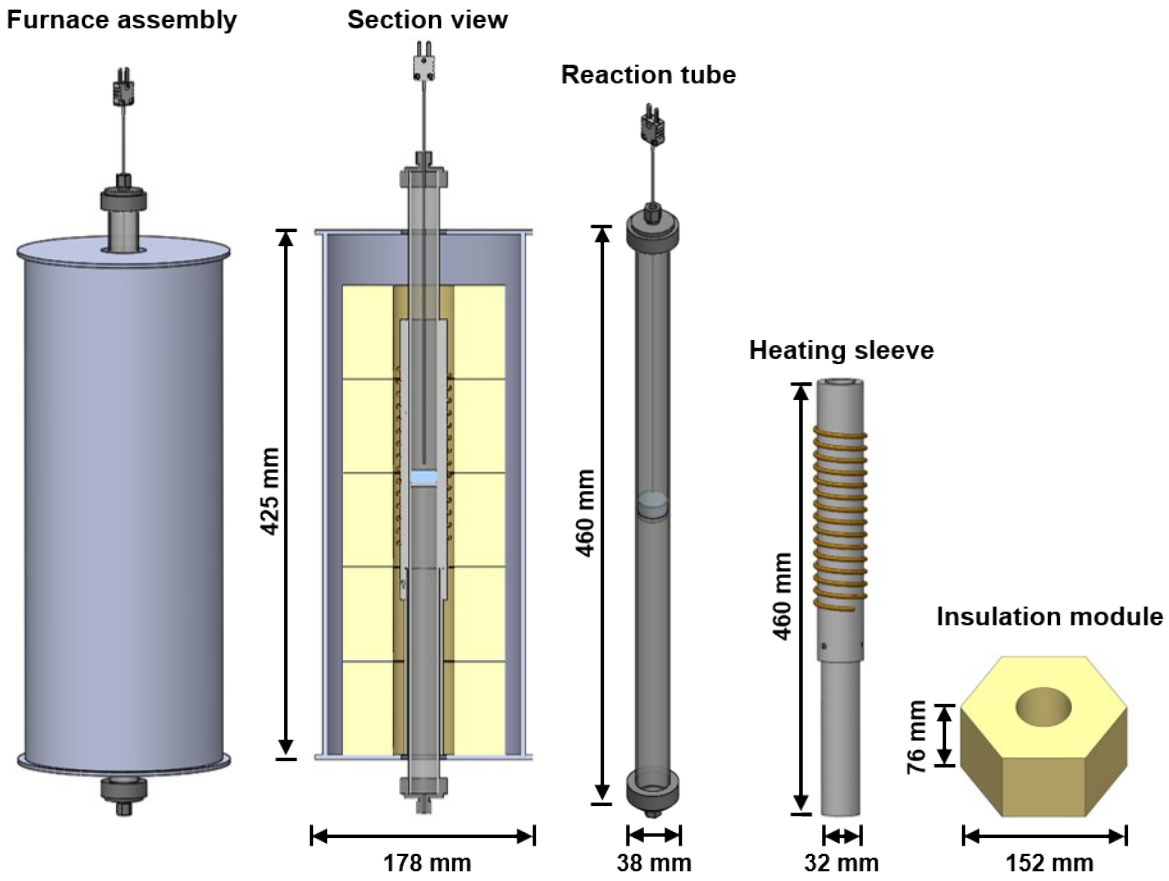


Figure S2. Furnace assembly and components with dimensions. The thermocouple controls the state (on or off) of the heater (brown coil) wrapped around the heating sleeve and is positioned 2 mm above the catalyst (blue) along the centerline of the reactor. The furnaces consisted of an external galvanized steel container (McMaster-Carr 1761K34), firebricks (McMaster-Carr 9355K2) as insulation, and a heating element. The firebricks were waterjet cut to size. The heating element and quartz reaction tube were placed in the center of the insulation. A custom-machined aluminum tube with a coil heater (Tempco Mightyband) wrapped around it was used as the heating element. A quartz tube [20 mm x 500 mm (ID x length)] modified with a frit (placed halfway down the tube; Technical Glass Products) to hold the catalyst was slipped through the aluminum tube.

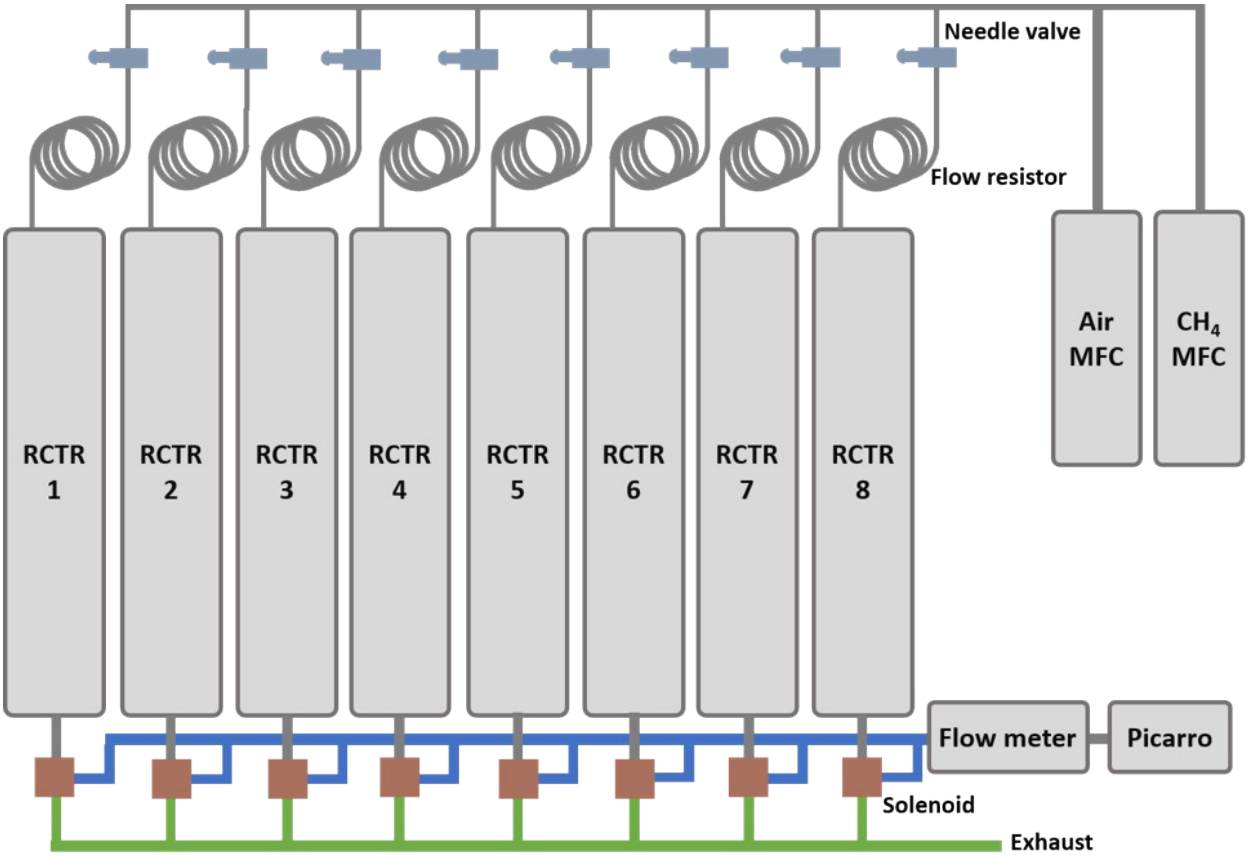


Figure S3. Schematic of flow control in CASHD. Mass flow controllers (MFCs) are supplied by upstream gas cylinders and the flow meter only measures the reactor that is being directed to the Picarro.

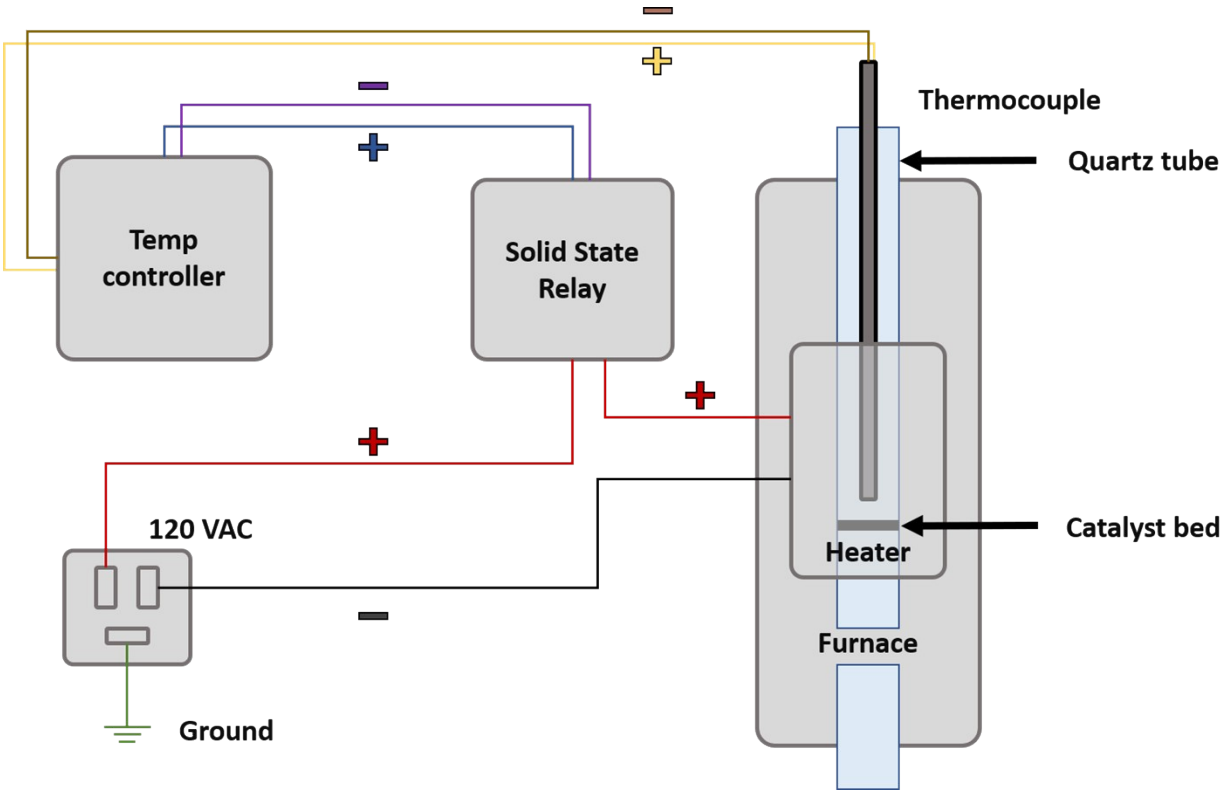


Figure S4. Reaction temperature control scheme. Each temperature controller was capable of controlling two reactors. The thermocouple was positioned 2 mm above the catalyst bed and a dry block calibrator (DwyerOmega DBCL-400-240) was used to verify the reported thermocouple specifications (±2.2 °C ±0.75% of the measured value).

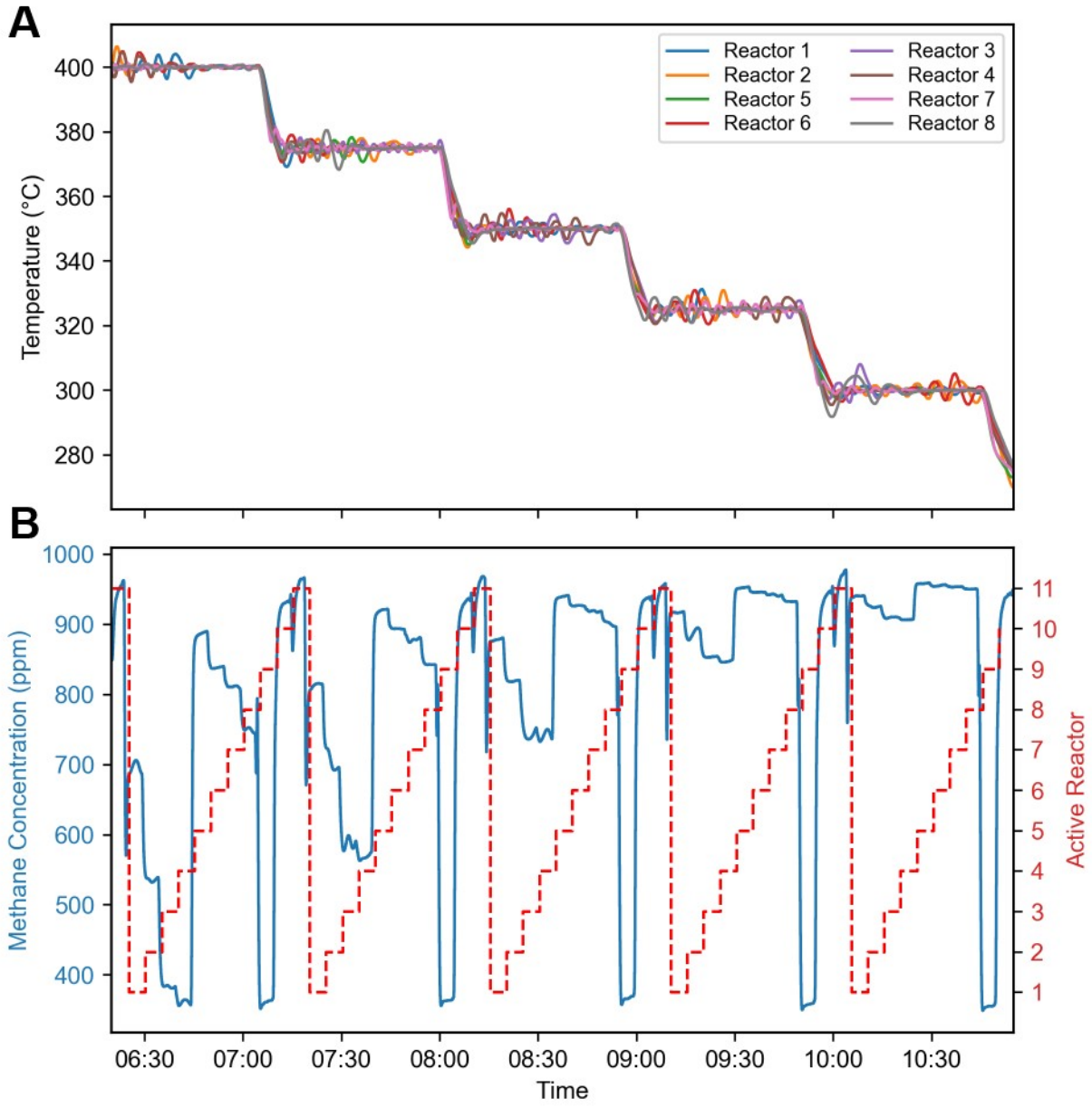


Figure S5. Demonstration of component synchronization. The (a) temperature profile and the (b) active reactors and measured concentration are all mapped to the same time profile for analysis. The temperature profile is timed to change setpoints when reactors 9 to 11 (reactors external to CASHD) are active. The active reactor signifies that the solenoid attached to that reactor is switched on and that the effluent is flowing to the measurement system.

Acid digestion and ICP-OES. The digestion of copper zeolite samples was conducted according to Galbraith Laboratories' procedure G-30C. It is available upon request and states:

"The sample is weighed into a suitable digestion tube (typically a 50 mL Class-A centrifuge tube) and is heated in an acidic solution in a PerkinElmer sample preparation block (SPB), or equivalent. H2O2 may be added to assist in the digestion of organic material. Once the digestion is complete, the solution is brought to Class A volume mark. The typical acids used include nitric (HNO3), hydrochloric (HCl), and hydrofluoric (HF)."

ICP-OES was conducted according to Galbraith Laboratories' procedure ME-70. It is available upon request and states:

"Samples and Quality Control Standards (QCS) are first prepared by an applicable preparation method into an acidified aqueous solution. An external calibration is created from analysis of prepared standards followed by analysis of blanks and various quality control solutions. Samples are then analyzed with bracketing control standards/blanks every 10 or less samples. Each sample matrix is examined for potential interference effects and as necessary may be re-analyzed using successive dilution, mathematically corrected for measured interferences, or analyzed with method of standard addition (MSA) to correct for these effects."

Table S3 Criterion for internal and external mass transfer

Catalyst	Temp (°C)	Temp (K)	observed rate (mol/g/s)[a]	rate (mol/s/m³)	order	Diffusivity of methane in air (m²/s)	Kinetmatic viscosity of air (m²/s)	Conc (mol/m³)	Volumetric flowrate (m³/s)	Superficial velocity (m/s)	Effective diffusivity of methane in pore (m²/s)	Re'	Sc	Sh'	k_c[b] (m/s)	Mears[c]	Weisz Prater[c]
1.9 BEA	390	663	1.6×10^{-8}	6.5×10^{-3}	0.75	8.8×10^{-5}	6.0×10^{-5}	0.021	3.7×10^{-5}	0.12	9.3×10^{-6}	1.63	0.69	1.13	0.30	1.9×10^{-4}	2.0×10^{-3}
2.5 BEA	391	664	2.8×10^{-8}	1.1×10^{-2}	0.77	8.8×10^{-5}	6.1×10^{-5}	0.021	3.7×10^{-5}	0.12	9.3×10^{-6}	1.63	0.69	1.13	0.30	3.4×10^{-4}	3.5×10^{-3}
3.4 BEA	390	663	3.6×10^{-8}	1.4×10^{-2}	0.75	8.8×10^{-5}	6.0×10^{-5}	0.021	3.7×10^{-5}	0.12	9.3×10^{-6}	1.63	0.69	1.13	0.30	4.3×10^{-4}	4.6×10^{-3}
5.0 BEA	390	664	6.4×10^{-8}	2.5×10^{-2}	0.76	8.8×10^{-5}	6.0×10^{-5}	0.021	3.7×10^{-5}	0.12	9.3×10^{-6}	1.63	0.69	1.13	0.30	7.7×10^{-4}	8.0×10^{-3}
1.8 FER	330	603	2.5×10^{-8}	1.0×10^{-2}	0.80	7.4×10^{-5}	5.2×10^{-5}	0.023	3.4×10^{-5}	0.11	7.9×10^{-6}	1.73	0.70	1.17	0.26	3.4×10^{-4}	3.4×10^{-3}
2.7 FER	330	603	4.2×10^{-8}	1.7×10^{-2}	0.75	7.4×10^{-5}	5.2×10^{-5}	0.023	3.4×10^{-5}	0.11	7.9×10^{-6}	1.73	0.70	1.17	0.26	5.3×10^{-4}	5.8×10^{-3}
3.7 FER	330	603	6.6×10^{-8}	2.6×10^{-2}	0.70	7.4×10^{-5}	5.2×10^{-5}	0.023	3.4×10^{-5}	0.11	7.9×10^{-6}	1.73	0.70	1.17	0.26	7.7×10^{-4}	9.0×10^{-3}
4.3 FER	330	603	7.0×10^{-8}	2.8×10^{-2}	0.70	7.4×10^{-5}	5.2×10^{-5}	0.023	3.4×10^{-5}	0.11	7.9×10^{-6}	1.73	0.70	1.17	0.26	8.2×10^{-4}	9.6×10^{-3}
1.9 MFI	330	604	5.1×10^{-8}	2.0×10^{-2}	0.88	7.4×10^{-5}	5.2×10^{-5}	0.023	3.4×10^{-5}	0.11	7.9×10^{-6}	1.73	0.70	1.17	0.26	7.4×10^{-4}	6.9×10^{-3}
2.5 MFI	330	603	8.0×10^{-8}	3.2×10^{-2}	0.78	7.4×10^{-5}	5.2×10^{-5}	0.023	3.4×10^{-5}	0.11	7.9×10^{-6}	1.73	0.70	1.17	0.26	1.0×10^{-3}	1.1×10^{-2}
3.5 MFI	305	578	6.1×10^{-8}	2.4×10^{-2}	0.75	6.9×10^{-5}	4.8×10^{-5}	0.024	3.2×10^{-5}	0.10	7.3×10^{-6}	1.78	0.70	1.18	0.24	7.7×10^{-4}	8.5×10^{-3}
4.3 MFI	306	579	7.5×10^{-8}	3.0×10^{-2}	0.71	6.9×10^{-5}	4.8×10^{-5}	0.024	3.2×10^{-5}	0.10	7.4×10^{-6}	1.78	0.70	1.18	0.24	8.9×10^{-4}	1.0×10^{-2}
1.9 MOR	350	623	4.0×10^{-8}	1.6×10^{-2}	0.78	7.9×10^{-5}	5.5×10^{-5}	0.023	3.5×10^{-5}	0.11	8.4×10^{-6}	1.69	0.70	1.15	0.27	5.1×10^{-4}	5.3×10^{-3}
2.7 MOR	350	623	6.5×10^{-8}	2.6×10^{-2}	0.76	7.8×10^{-5}	5.5×10^{-5}	0.023	3.5×10^{-5}	0.11	8.4×10^{-6}	1.69	0.70	1.15	0.27	8.1×10^{-4}	8.6×10^{-3}
3.3 MOR	350	623	8.0×10^{-8}	3.2×10^{-2}	0.72	7.8×10^{-5}	5.5×10^{-5}	0.023	3.5×10^{-5}	0.11	8.4×10^{-6}	1.69	0.70	1.15	0.27	9.4×10^{-4}	1.1×10^{-2}
5.0 MOR	351	624	9.5×10^{-8}	3.8×10^{-2}	0.70	7.9×10^{-5}	5.5×10^{-5}	0.023	3.5×10^{-5}	0.11	8.4×10^{-6}	1.69	0.70	1.15	0.27	1.1×10^{-3}	1.2×10^{-2}

a. The rate at the highest measured temp was chosen for this analysis because this would be the most likely condition to have mass tranfer limitations

b. Mass transfer coefficient calculated with Thoenes-Kramer relationship

c. Mears number of < 0.15 and a Weisz Prater << 1 suggest internal and external mass transfer are not affecting the measured kinetics

Table S2 Weisz-Prater and Mears criterion definition

Weisz-Prater criterion	$WP = \frac{r_{obs} R_p^2}{C_{CH_4,s} D_e}$
Mears Criterio	
Variable descript	
Pellet radius	
Methane concentra surface	
Effective diffusiv	
Observed rate	
Powder bulk den	
Mass transfer coef	
Methane bulk conce	

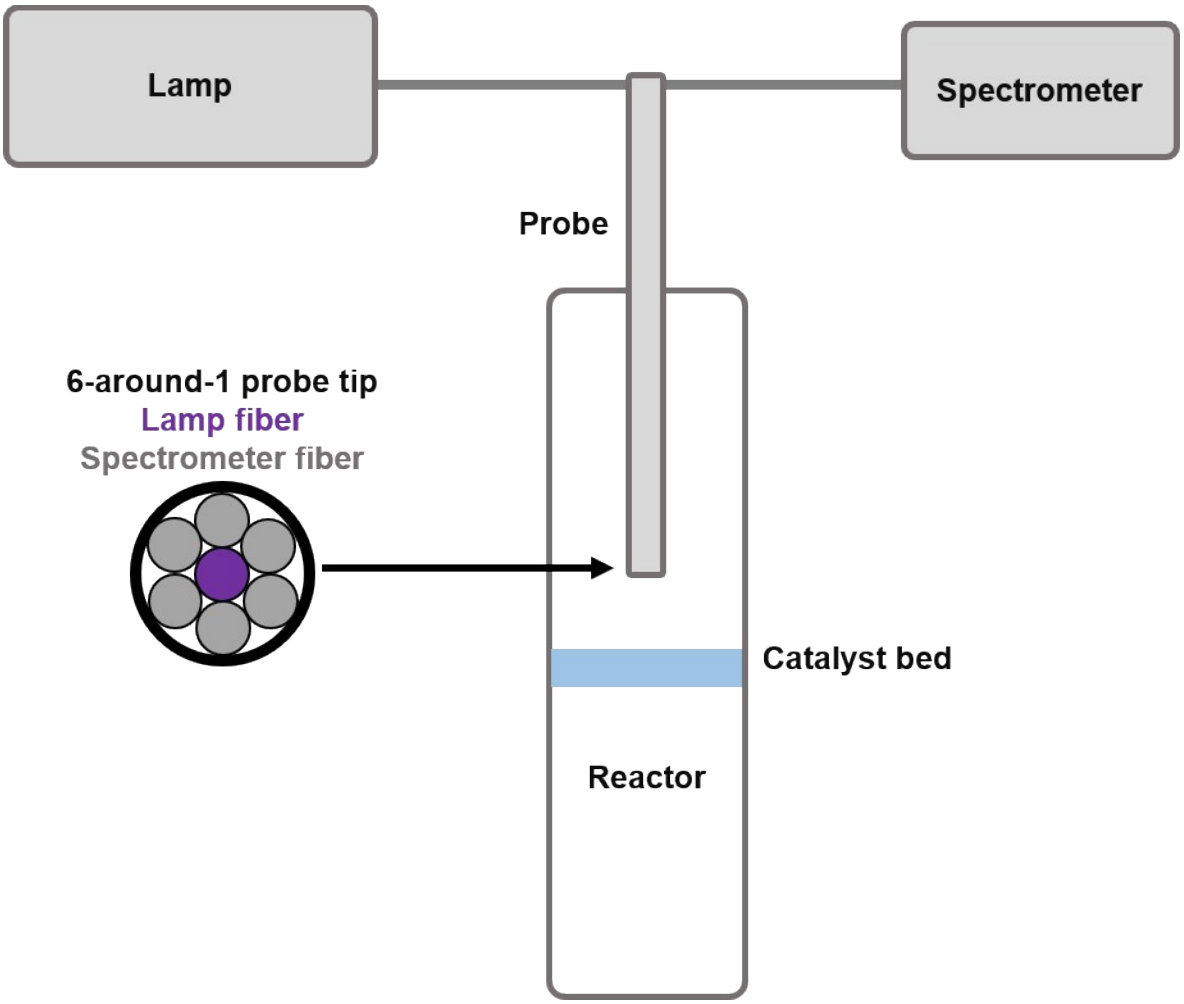
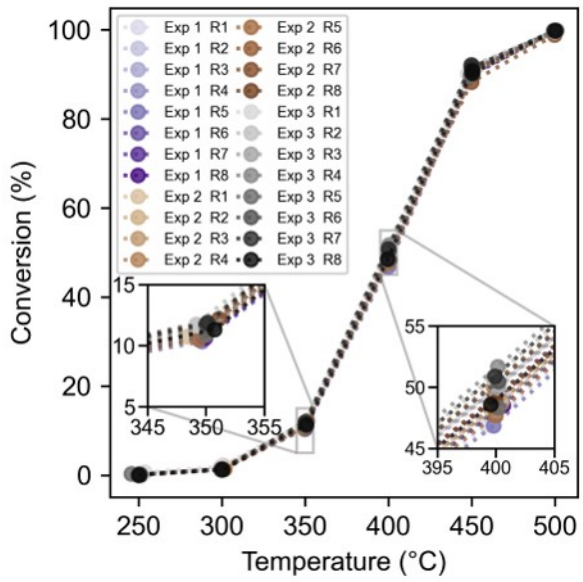


Figure S6. Schematic of in situ diffuse reflectance ultraviolet-visible spectroscopy set up.

Table S4 Variance components of CASHD from benchmarking experiment

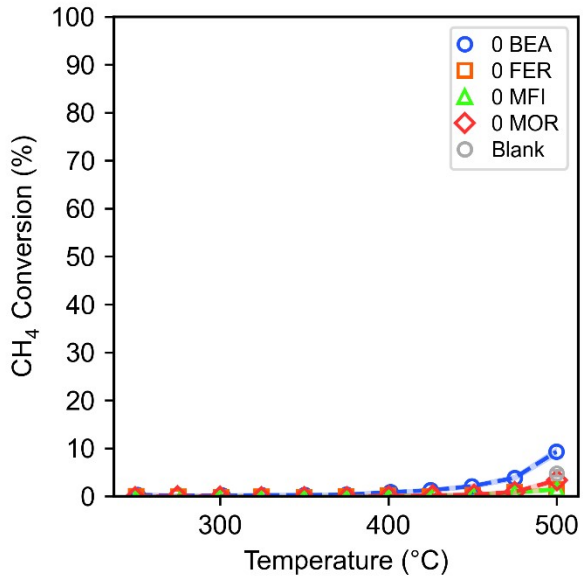
Temp (°C)	Conversion (%)	$\sigma_{reactor}$	$\sigma_{experiment}$	$\sigma_{residual}$
300	1.45	0.063	0.003	0.005
350	11.21	0.093	0.127	0.071
400	49.18	0.647	0.650	0.460
450	90.70	0.072	0.110	0.500
500	99.70	0.004	0.003	0.038



67

68 **Figure S7.** Benchmarking experiment for CASHD uncertainty envelope. Light-off curves of the

69 same catalyst in each reactor (R1-R8) tested three times (Exp. 1 to Exp. 3).



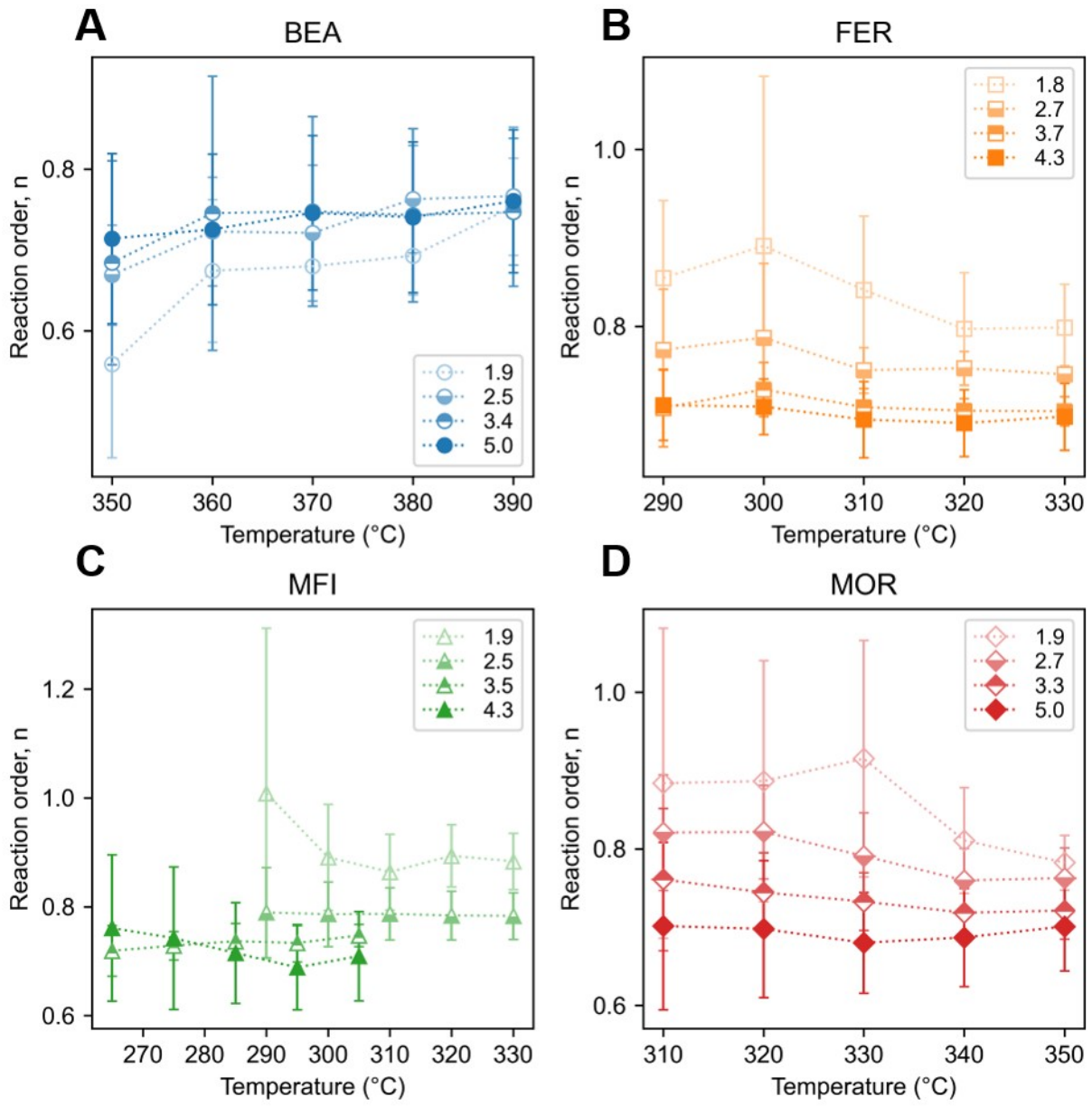
70

71 **Figure S8.** Methane oxidation light-off curves for blank zeolites. Frameworks BEA, FER, MFI,

72 and MOR were evaluated, where the number zero in the legend indicates zero copper loading

73 (wt. %). The average conversion of four empty reaction tubes at 500 °C and the associated

74 standard deviation is shown and denoted in the legend as Blank. Experiments were run at 1000
75 ppm methane in dry air at 100 SCCM over 500 mg of zeolite. Dotted lines are to guide the eye;
76 shading represents the uncertainty envelope.



77

78 **Figure S9.** Methane reaction order across temperatures Frameworks (a) BEA, (b) FER, (c) MFI,
79 and (d) MOR were evaluated, where the numbers in the legend correspond to measured copper

loading (wt. %). The reaction order was determined from an order experiment where methane was varied from 12 to 1200 ppm for each temperature tested in the Arrhenius experiments.

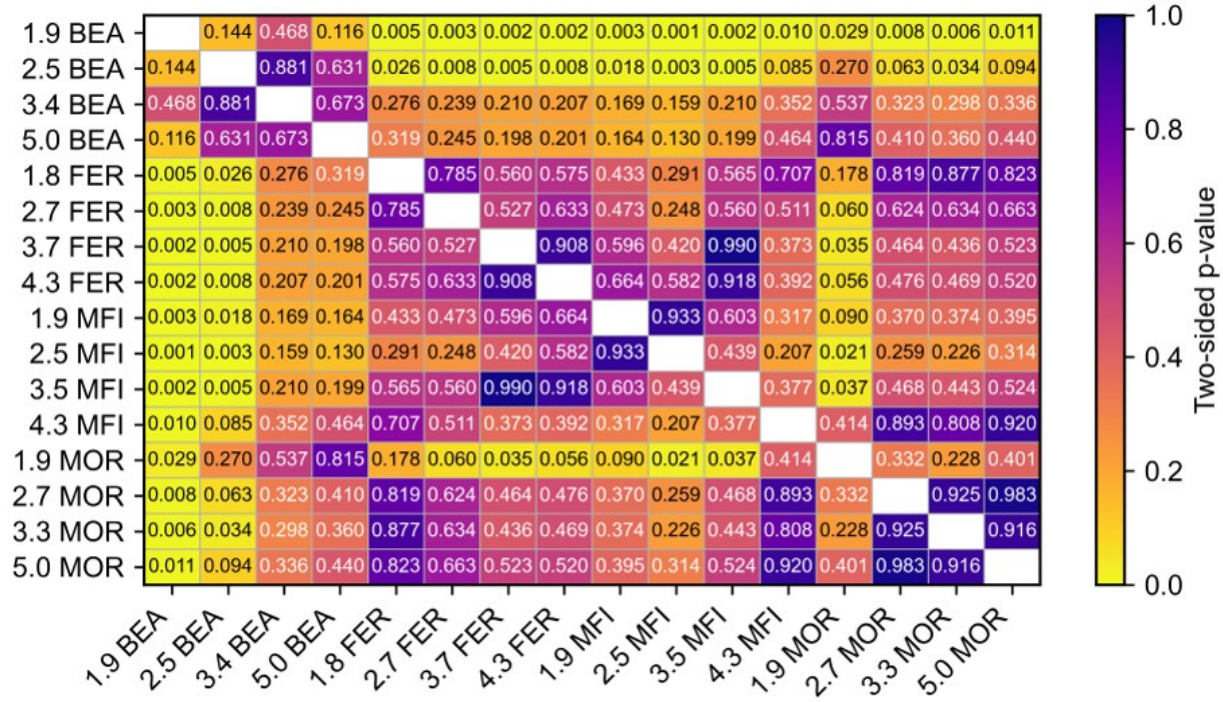
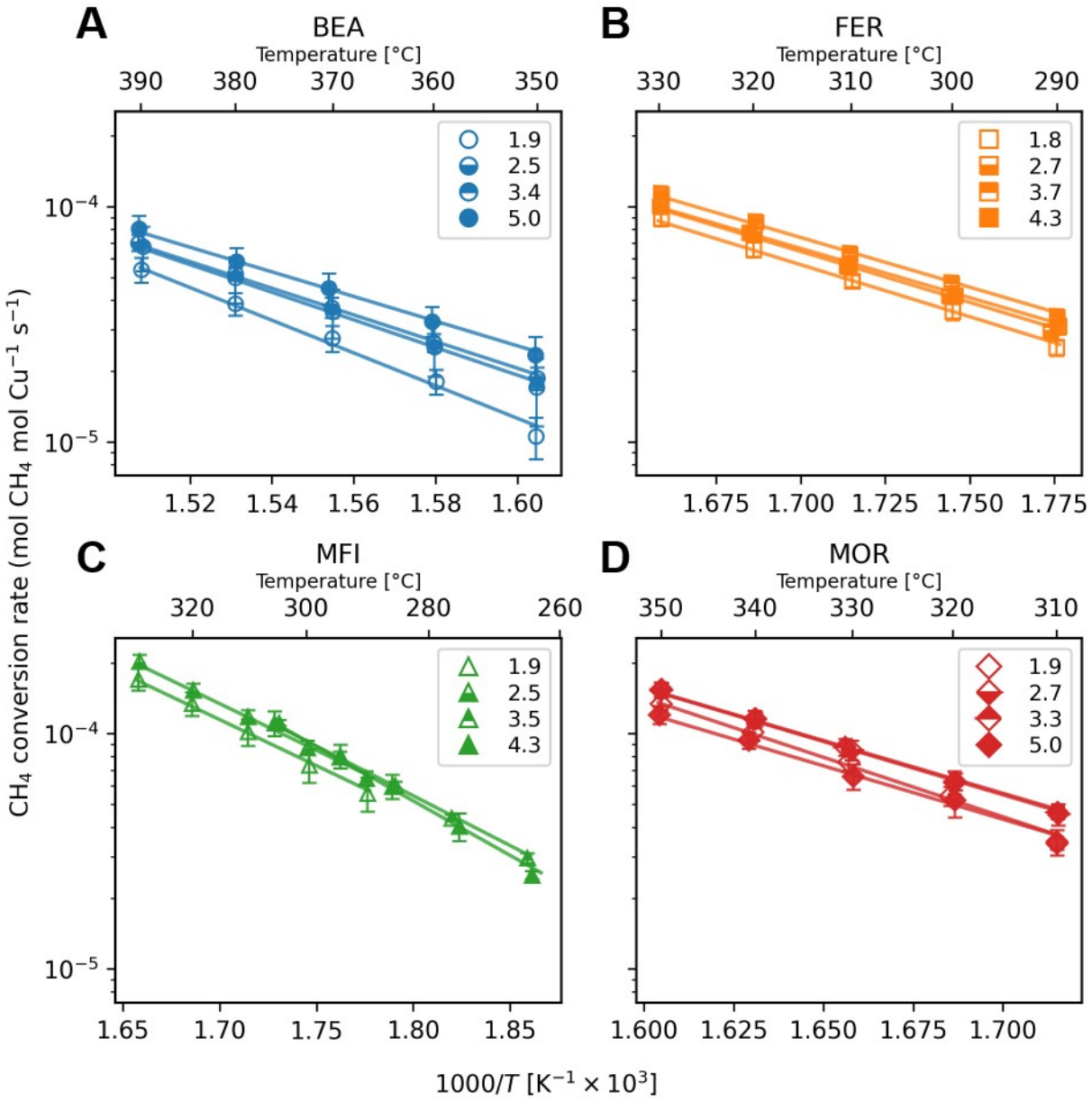


Figure S10. Statistical significance of each catalyst's activation energy. P-values are displayed inside each cell filled with the corresponding color intensity.



85

86 **Figure S11.** Arrhenius plots of all catalysts tested. Frameworks (a) BEA, (b) FER, (c) MFI, and

87 (d) MOR were evaluated, where the numbers in the legend correspond to measured copper

88 loading (wt. %).

89

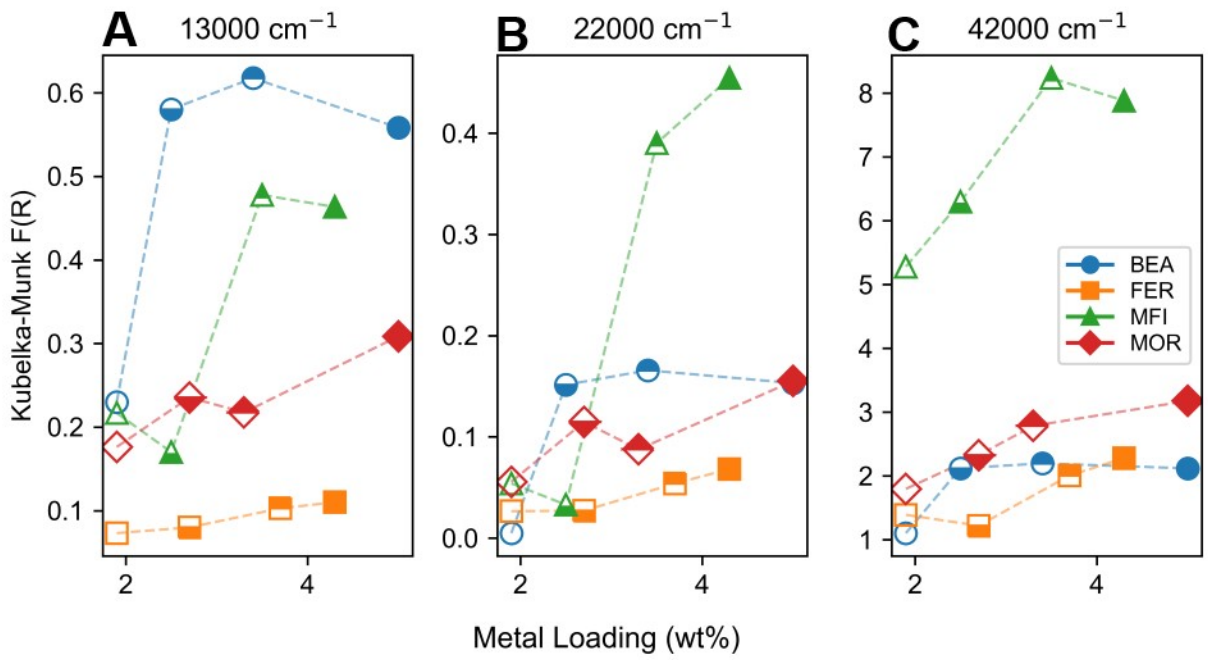


Figure S12. Kubelka-Munk absorbance at (a) 13000 cm^{-1}, (b) 22000 cm^{-1}, and 42000 cm^{-1} for various copper-doped zeolites. Kubelka-Munk absorbance is given by $F(R) = (1-R)^2/(2R)$.

Temperature control program. The setpoint of the Omega CN-402 controllers were updated remotely with this code.

```
import os
import minimalmodbus
import time
import csv
from threading import Thread, Lock

# Global lock for synchronizing access to shared serial ports
serial_lock = Lock()
```

```
104
105 # --- Configuration Constants ---
106 LOGGING_INTERVAL = 10  # Seconds
107 BAUDRATE = 19200
108 TIMEOUT = 2.0
109
110 # --- Helper Functions ---
111
112 def log_data(filename, data):
113     with open(filename, 'a', newline='') as file:
114         writer = csv.writer(file)
115         writer.writerow(data)
116
117 def log_error(filename, message):
118     timestamp = time.strftime('%Y-%m-%d %H:%M:%S', time.localtime())
119     with open(filename, 'a', newline='') as file:
120         writer = csv.writer(file)
121         writer.writerow([timestamp, message])
122
123 def read_register_safe(instrument, address, retries=3, delay=1):
124     for i in range(retries):
125         with serial_lock:
126             try:
```

```
            # Function code 3, 0 decimals
            return instrument.read_register(address, 0, 3)
        except IOError:
            if i < retries - 1:
                time.sleep(delay)
            else:
                raise

def write_register_safe(instrument, address, value, retries=3, delay=1):
    for i in range(retries):
        with serial_lock:
            try:
                instrument.write_register(address, int(value), 0)
                return
            except IOError:
                if i < retries - 1:
                    time.sleep(delay)
                else:
                    raise

# --- Core Control Logic ---

```

```
149 def run_temperature_program(instrument, program_steps, log_filename, pv_addr, sp_addr,
150 loop_num):
151     """
152     Executes a sequence of ramp and hold steps for a specific reactor loop.
153     """
154     error_log = log_filename.replace('.csv', '_errors.csv')
155 
156     # Initialize log file
157     if not os.path.isfile(log_filename):
158         with open(log_filename, 'w', newline='') as file:
159             writer = csv.writer(file)
160             writer.writerow(['Time', 'PV Temperature (°C)', 'Setpoint (°C)', 'Loop ID'])
161 
162     for step in program_steps:
163         step_type = step["type"]
164 
165         if step_type == "ramp":
166             target = float(step["target"])
167             rate = float(step["rate"])
168 
169             try:
170                 start_sp = float(read_register_safe(instrument, sp_addr))
171             except IOError as e:
```

```
            log_error(error_log, f"Init ramp read error: {e}")
            continue

        start_time = time.time()
        last_log = start_time

        while True:
            elapsed = time.time() - start_time

            # Calculate soft setpoint
            if start_sp < target:
                current_sp = min(start_sp + (rate * elapsed / 60.0), target)
            elif start_sp > target:
                current_sp = max(start_sp - (rate * elapsed / 60.0), target)
            else:
                current_sp = target

            # Update controller setpoint
            try:
                write_register_safe(instrument, sp_addr, int(current_sp))
            except IOError as e:
                log_error(error_log, f"Ramp write error: {e}")

```

```
            # Logging
            if time.time() - last_log >= LOGGING_INTERVAL:
                try:
                    pv = read_register_safe(instrument, pv_addr) / 10.0
                    log_data(log_filename, [
                        time.strftime('%Y-%m-%d %H:%M:%S'),
                        f"{pv:.1f}", f"{current_sp:.1f}", loop_num
                    ])
                except IOError as e:
                    log_error(error_log, f"Ramp PV read error: {e}")
                last_log = time.time()

            if abs(current_sp - target) < 0.1:
                break

            time.sleep(1)

        elif step_type == "hold":
            temp = float(step["temperature"])
            duration = float(step["duration"])

            try:
                write_register_safe(instrument, sp_addr, int(temp))
```

```
        except IOError as e:
            log_error(error_log, f"Hold set error: {e}")

        start_time = time.time()
        last_log = start_time

        while (time.time() - start_time) < duration:
            if time.time() - last_log >= LOGGING_INTERVAL:
                try:
                    pv = read_register_safe(instrument, pv_addr) / 10.0
                    real_sp = float(read_register_safe(instrument, sp_addr))
                    log_data(log_filename, [
                        time.strftime('%Y-%m-%d %H:%M:%S'),
                        f"{pv:.1f}", f"{real_sp:.1f}", loop_num
                    ])
                except IOError as e:
                    log_error(error_log, f"Hold PV read error: {e}")
                last_log = time.time()

            time.sleep(1)

# --- Experimental Parameters ---

```

```
# Define durations (seconds)
T_CALCINATION = 5 * 3600
T_REACTION = 55 * 60

# Define temperature profile (Celsius)
# Sequence: Calcination -> Series of descending reaction temperatures
profile_temps = [
    (500, T_CALCINATION), (500, T_REACTION), (475, T_REACTION),
    (450, T_REACTION), (425, T_REACTION), (400, T_REACTION),
    (375, T_REACTION), (350, T_REACTION), (325, T_REACTION),
    (300, T_REACTION), (275, T_REACTION), (250, T_REACTION),
    (0, T_REACTION)
]

# Construct the program steps list
program_sequence = [{"type": "hold", "temperature": 450.0, "duration": 2}] # Initial check
for temp, duration in profile_temps:
    program_sequence.append({"type": "hold", "temperature": float(temp), "duration":
float(duration)})

# Controller Configuration
# Note: Address assignment assumes 2 loops per controller
controllers = [
```

```
    {"port": "COM6", "address": 247, "log": "temp_log_com6.csv"},
    {"port": "COM16", "address": 246, "log": "temp_log_com16.csv"},
    {"port": "COM15", "address": 245, "log": "temp_log_com15.csv"},
    {"port": "COM18", "address": 244, "log": "temp_log_com18.csv"}
]

# --- Main Execution ---

if __name__ == "__main__":
    threads = []

    # Initialize instruments and threads
    for config in controllers:
        try:
            inst = minimalmodbus.Instrument(config["port"], config["address"])
            inst.serial.baudrate = BAUDRATE
            inst.serial.timeout = TIMEOUT
            inst.serial.parity = minimalmodbus.serial.PARITY_NONE

            # Configure Loop 1 and Loop 2 for each controller
            loop_configs = [
                {"pv": 1000, "sp": 1300, "id": 1},
                {"pv": 1001, "sp": 1304, "id": 2}
```

```
            ]

            for loop in loop_configs:
                t = Thread(target=run_temperature_program, args=(
                    inst,
                    program_sequence,
                    config["log"],
                    loop["pv"],
                    loop["sp"],
                    loop["id"]
                ))
                threads.append(t)
                t.start()
                time.sleep(0.2) # Stagger start to ease serial load

        except Exception as e:
            print(f"Initialization failed for {config['port']}: {e}")

    # Wait for completion
    for t in threads:
        t.join()
```