Supplementary Information

Insight into charged drug release from metal-organic frameworks

Josh Phipps,[a] Luis Pinzon-Hererra,[b] Wenlu Fan,[a] Sharmila Neupane,[a] Paige Walsh,[a] Abby Tian,[a] Teddy Biggin,[a] Jorge Almodovar,[b] Thamraa AlShahrani,[c] and Shengqian Ma*[a]

[a]Department of Chemistry, University of North Texas, Denton, Texas 76205, United States
[b]Department of Chemical, Biochemical, and Environmental Engineering, University of Maryland Baltimore County, Baltimore, Maryland, 21250, United States
[c]Department of Physics, College of Science, Princess Nourah bint Abdulrahman University, Riyadh 11564, Saudi Arabia

Table of Contents

Materials and Methods

Chemicals

All chemicals were purchased from suppliers and used as is without further purification: 2-aminoterephthalic acid from Thermo Scientific (CAS: 10312-55-7), 1,3,5-benzenetricarboxylic acid from Alfa Aesar (CAS: 554-95-0), 4-((4-(dimethylamino)phenyl)diazenyl)benzenesulfonic acid sodium salt (methyl orange) from AmBeed (CAS: 547-58-0), epinephrine from Sigma Aldrich (CAS: 51-43-4), heparin sodium from AA Blocks (CAS: 9041-08-1), 2-hydroxyterephthalic acid from AmBeed (CAS: 636-94-2), iron(III) chloride hexahydrate from Alfa Aesar (CAS: 10025-77-1), methylene blue hydrate from TCI (CAS: 122965-43-9), nitroterephthalic acid from Acros Organics (CAS: 610-29-7), penicillin G sodium salt (CAS: 69-57-8), poly(ethyleneimine) solution from Aldrich (CAS: 9002-98-6, 50 wt. % in H2O), terephthalic acid from Acros Organics (CAS: 100-21-0), and zirconium (IV) chloride from Aldrich (CAS: 10026-11-6).

Instrumentation

Characterization and experimental testing were conducted using the following instruments: Powder X-ray Diffraction (pXRD) patterns were collected using a Rigaku SmartLab SE, morphology was observed using a JEOL JSM-IT2000 scanning electron microscope, Brauner-Emmet-Teller (BET) surface area was determined using Micromeritics ASAP 2020 Plus surface analyzer, thermogravimetric analysis (TGA) was performed using a Mettler Toledo Thermal Analysis System TGA/DSC 3+. Infrared spectroscopy was performed using a Bruker Vertex 70, UV-Visible spectrum absorption measurements were performed using a JASCO V-670 Spectrophotometer, and zeta potential was determined using a Zetasizer Nano ZS90.

Software

Microsoft® Excel® 2019 MSO was used to analyse results, graphs were produced using OriginPro® 2022 SR1, and MATLAB® R2023b was utilized for calculations and plotting fitted curves.

MOF Preparation

UiO-66 and its derivatives were synthesized according to a report by He et al.[1] with minor modifications described below:

UiO-66 Synthesis

In a 500 ml two-neck round bottom flask, 2.68g (16.1 mmol) $ZrCl_4$, 2.67g (16.1 mmol) Terephthalic acid and 110.5 mL (1.93 mol) acetic acid were dissolved in 250 mL dimethylformamide (DMF). Then, 18.75 mL H_2O was added. The solution was then heated in an oil bath stirring at 400 rpm and was kept at 120 °C for 15 min before being cooled to room temperature. The product was collected by centrifugation at 5,000 rpm for 10 minutes and the supernatant was discarded prior to activation.

UiO-66-NH2 Synthesis

In a 500 ml two-neck round bottom flask, 3.75 g (16.1 mmol) $ZrCl_4$, 2.92 g (16.1 mmol) 2-Aminoterephthalic acid and 110.5 mL (1.93 mol) acetic acid were dissolved in 250 mL DMF. Then, 18.75 mL H_2O was added. The solution was then heated in an oil bath stirring at 400 rpm and was kept at 120 °C for 15 min before being cooled to room temperature. The product was collected by centrifugation at 5,000 rpm for 10 minutes and the supernatant was discarded prior to activation.

UiO-66-OH Synthesis

In a 500 ml two-neck round bottom flask, 3.75 g (16.1 mmol) $ZrCl_4$, 2.93 g (16.1 mmol) 2-Hydroxyterephthalic acid and 110.5 mL (1.93 mol) acetic acid were dissolved in 250 mL DMF. Then 18.75 mL H_2O was added. The solution was then heated in an oil bath stirring at 400 rpm and was kept at 120 °C for 15 min before being cooled to room temperature. The product was collected by centrifugation at 5,000 rpm for 10 minutes and the supernatant was discarded prior to activation.

UiO-66-NO2 Synthesis

In a 500 ml two-neck round bottom flask, 3.75 g (16.1 mmol) $ZrCl_4$, 3.40g (16.1 mmol) 2-Nitroterephthalic acid and 110.5 mL (1.93 mol) acetic acid were dissolved in 250 mL DMF. Then 18.75 mL H_2O was added. The solution was then heated in an oil bath stirring at 400 rpm and was kept at 120°C for 15 min before being cooled to room temperature. The product was collected by centrifugation at 5,000 rpm for 10 minutes and the supernatant was discarded prior to activation.

MIL-100(Fe) Synthesis

MIL-100(Fe) synthesis was conducted following a report by Zhang et al.[2] with minor modifications. In a two-neck 25 mL round bottom flask, $Fe(NO_3)_3 \cdot 9H_2O$ (4.04 g, 10 mmol), H_3BTC (1.89 g, 9 mmol), and deionized water (6 mL) were combined and stirred. The solution was brought to reflux at 95°C was allowed to react overnight. The product was then collected by centrifugation at 5,000 rpm for 10 minutes. The solid was then washed three times each with deionized water and ethanol at 70 ºC for 24 hrs before being isolated again via centrifuge and dried in a vacuum desiccator at 150°C overnight to activate.

MOF Activation

For each MOF synthesized in DMF, the solid product was placed in round bottom flask with a reflux condenser attached, fresh DMF was added, and the mixture was stirred at 120°C for 3-5 hours before allowing the mixture to cool to room temperature. The mixture was then centrifuged to remove the supernatant and placed in fresh DMF overnight for solvent exchange. This was repeated for a total of 3 cycles. Next, the same process was repeated at 60°C in acetone following the same procedure. Finally, the dry MOFs were collected and placed in a 120°C vacuum oven overnight to complete activation.

Size Exclusion of MOFs

A sample of 100 mg from MOF sample was suspended with 10 mL of Acetone inside a 15 mL centrifuge tube. The solution was sonicated for 10 minutes before being centrifuged at 1000 rpm for 5 minutes. The opaque suspension was recovered while the pellet was returned to the stock vial of MOF. Next, the solution was centrifuged at 2,500 rpm for 5 minutes and the supernatant

was removed and returned to the stock vial. The remaining MOF pellet of size excluded (SE) MOFs was recovered. This process was repeated until at least 50 mg was recovered. The particle size of each MOF was then measured using a Microtrac NANOTRAC Flex particle size analyzer before each sample was dried overnight at 120 °C in a vaccum oven and used for further testing.

Stability Test

In two 15 mL centrifuge tubes, 30 mg of size excluded MIL-100(Fe) was added and suspended in 10 mL of 20mM HEPES solution buffered at pH 7. The solution was vortexed and mixed thoroughly and allowed to sit for 10 minutes. After the allotted time, the solution was centrifuged for 5 minutes at 6,000 rpm and the supernatant was removed. The solid was then washed 3 times with DI water and 3 times with acetone before being placed in a vacuum oven at 50 °C to dry for 6 hours. This was repeated with a solution of 1% PEI buffered at pH 7 with 20 mM HEPES before both were examined using pXRD.

Cell Culture

Human mesenchymal stem cells (hMSCs) derived from bone marrow were obtained from RoosterBio (Cat. #MSC-003) and used at passage 5. The donor was a healthy 25-year-old male (Lot #00174). According to the vendor's product specification sheet, these cells demonstrated the capacity to differentiate into both osteogenic and adipogenic lineages. Cells were cultured in Minimum essential medium alpha (MEM Alpha, 1X; Gibco, Cat. #12561-056), supplemented with 1.2% penicillin-streptomycin (Corning, Cat. #30002CI), and 1.2% L-glutamine (Corning, Cat. #25005CI). The medium was further enriched with 20% fetal bovine serum (Gibco, Cat. #12662029) prior to the use with cells. Cultures were maintained in a humidified incubator at 37 °C with 5% CO_2. The culture medium was replaced every 48 hours, and experiments were conducted when cells reached approximately 80% confluence.

hMSC Viability:

To assess hMSCs viability, a PrestoBlueTM HS Cell Viability Reagent (Invitrogen, Cat. #P50200) was employed. hMSCs were seeded at a density of 25,000 cells/cm² in 96-well plates and maintained for three days under control and experimental conditions. The viability assay followed protocols previously established in our laboratory and described in our previous works [1-4]. After the culture period, the medium was aspirated and replaced with 100 µL of a solution composed of 90% fresh culture medium and 10% PrestoBlueTM HS reagent. Plates were incubated for 3 hours, after which fluorescence was measured at 560 nm excitation and 590 nm emission using a Synergy LX FA multi-mode microplate reader (Agilent BioTek Cat. No. BTSLXFA) with a red filter cube (EX 530/25, EM 590/25, Cat. No. BT1505004). Results are presented as the mean ± standard deviation from five technical replicates per condition.

In addition, bright-field images were captured after three days of culture using a MOTIC Led Inverted Microscope (Ref. AE31TR) to qualitatively assess cell adhesion and morphology. These observations allowed for the identification of noticeable differences in cell behavior when compared to tissue culture plastic (control) conditions. Data are expressed as the mean ± standard error of the mean. Statistical comparisons between multiple groups were conducted using one-way analysis of variance (ANOVA). Differences were considered statistically significant when the p-value was less than 0.05.

MOF Characterization

Powder X-ray Diffraction (pXRD):

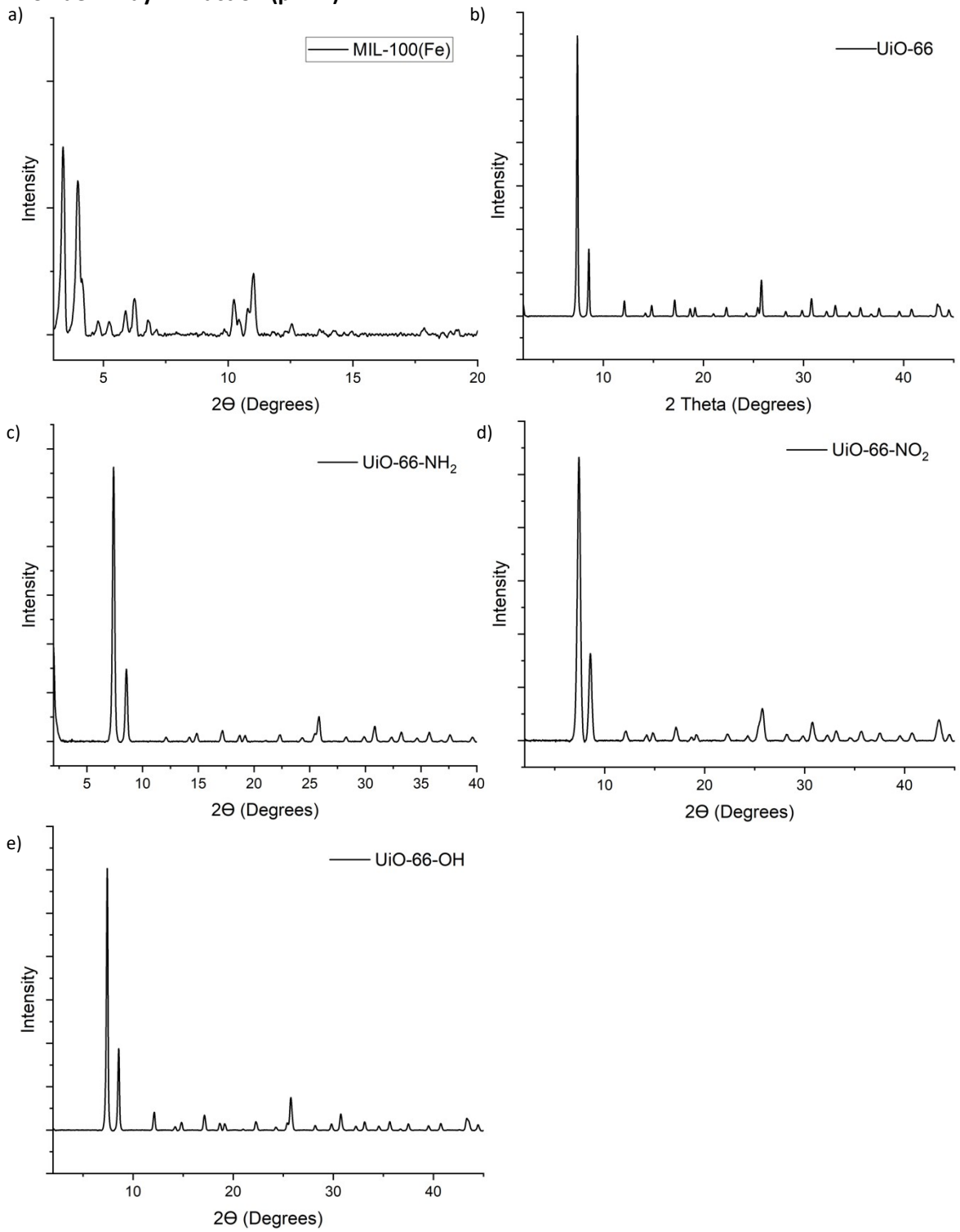


Fig. S1: PXRD patterns of synthesized MOFs (a) MIL-100(Fe), (b) UiO-66, (c) UiO-66-NH_2, (d) UiO-66-NO_2, and (e) UiO-66-OH.

Brunaeur-Emmett-Teller (BET) Absorption:

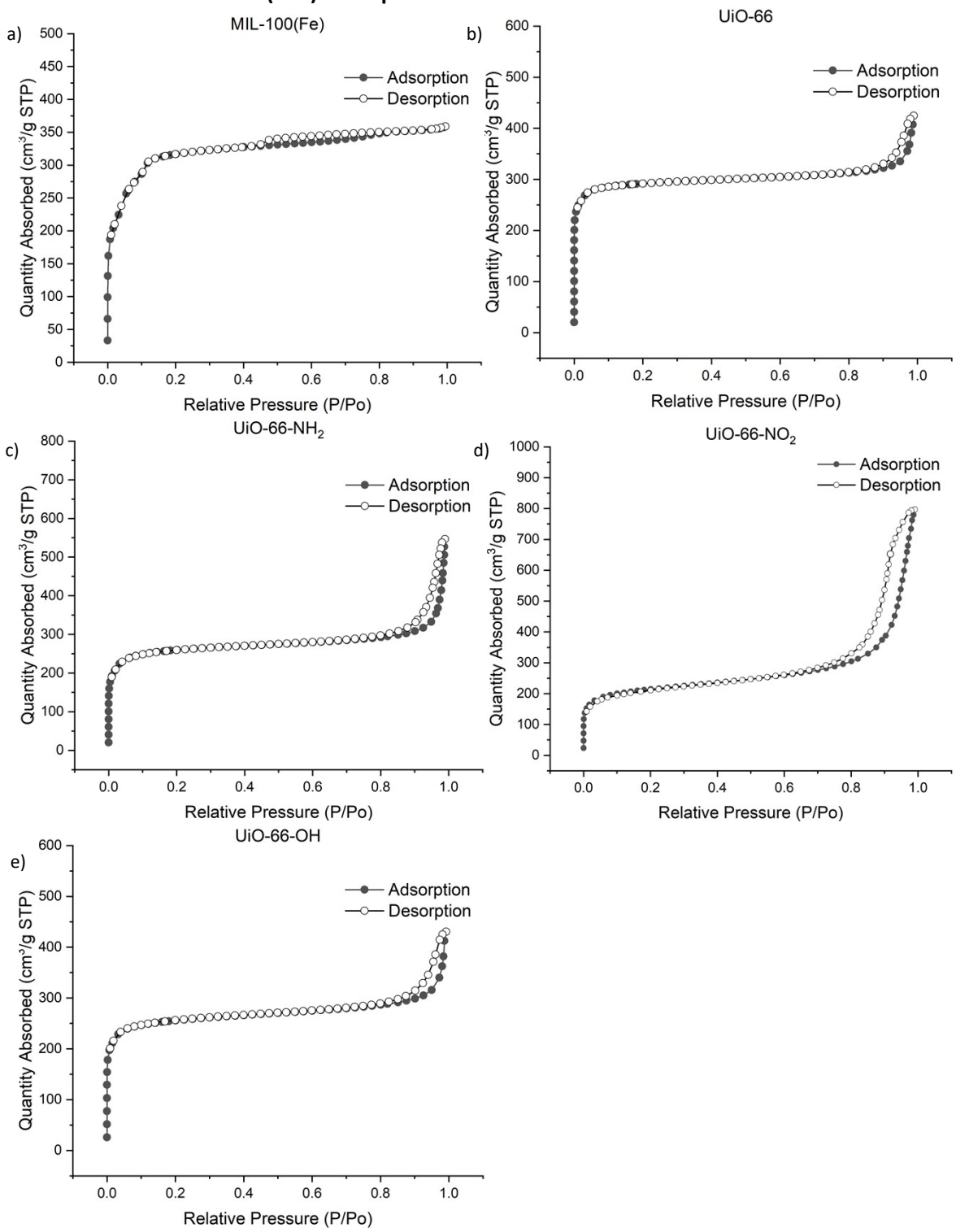


Fig. S2: BET isotherms from synthesized MOFs (a) MIL-100(Fe), (b) UiO-66, (c) UiO-66-NH2, (d) UiO-66-NO2, and (e) UiO-66-OH.

Pore Width Analysis:

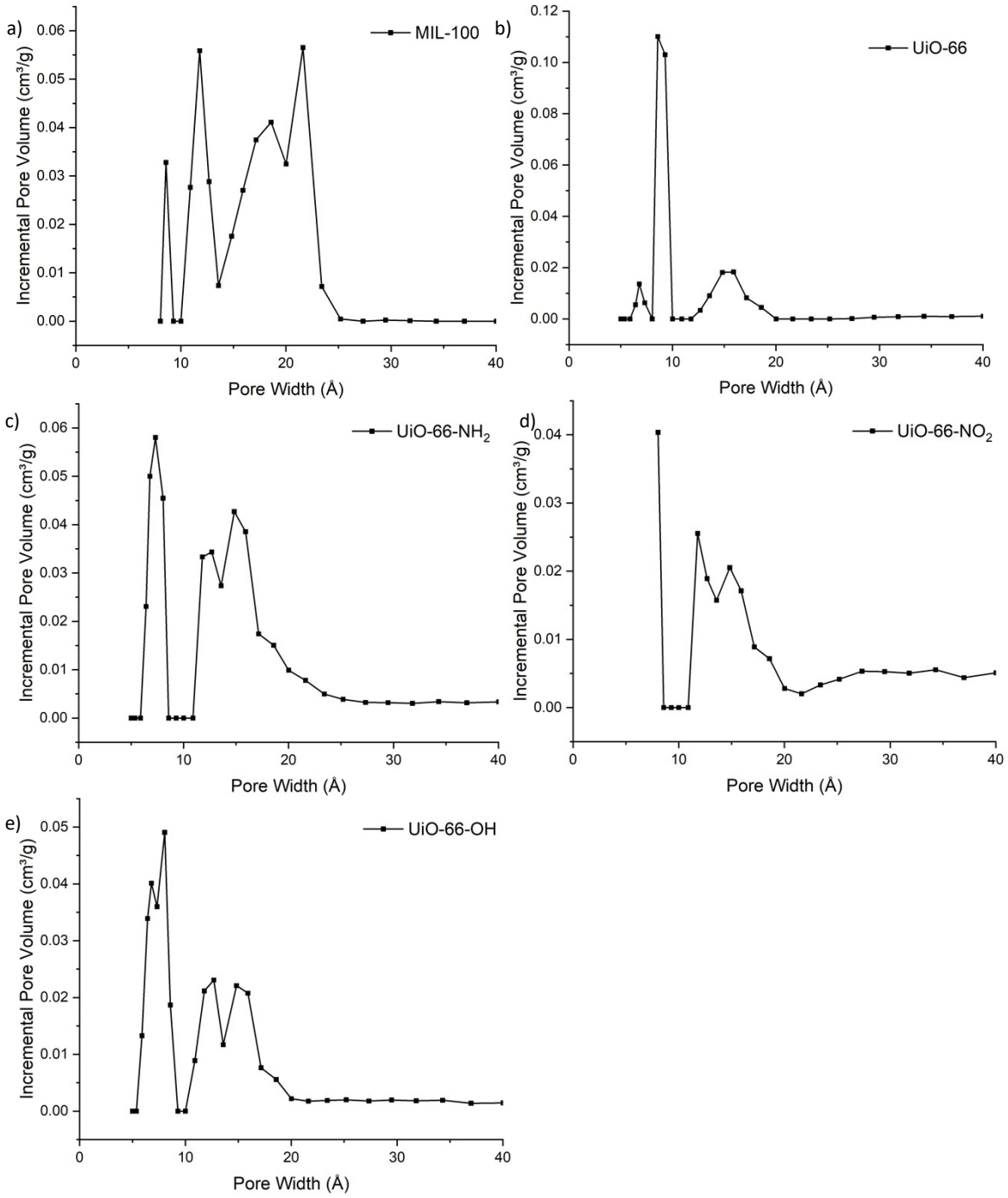


Fig. S3: Pore width plots obtained from synthesized MOFs (a) MIL-100(Fe), (b) UiO-66, (c) UiO-66-NH2, (d) UiO-66-NO2, and (e) UiO-66-OH.

Thermogravimetric Analysis (TGA):

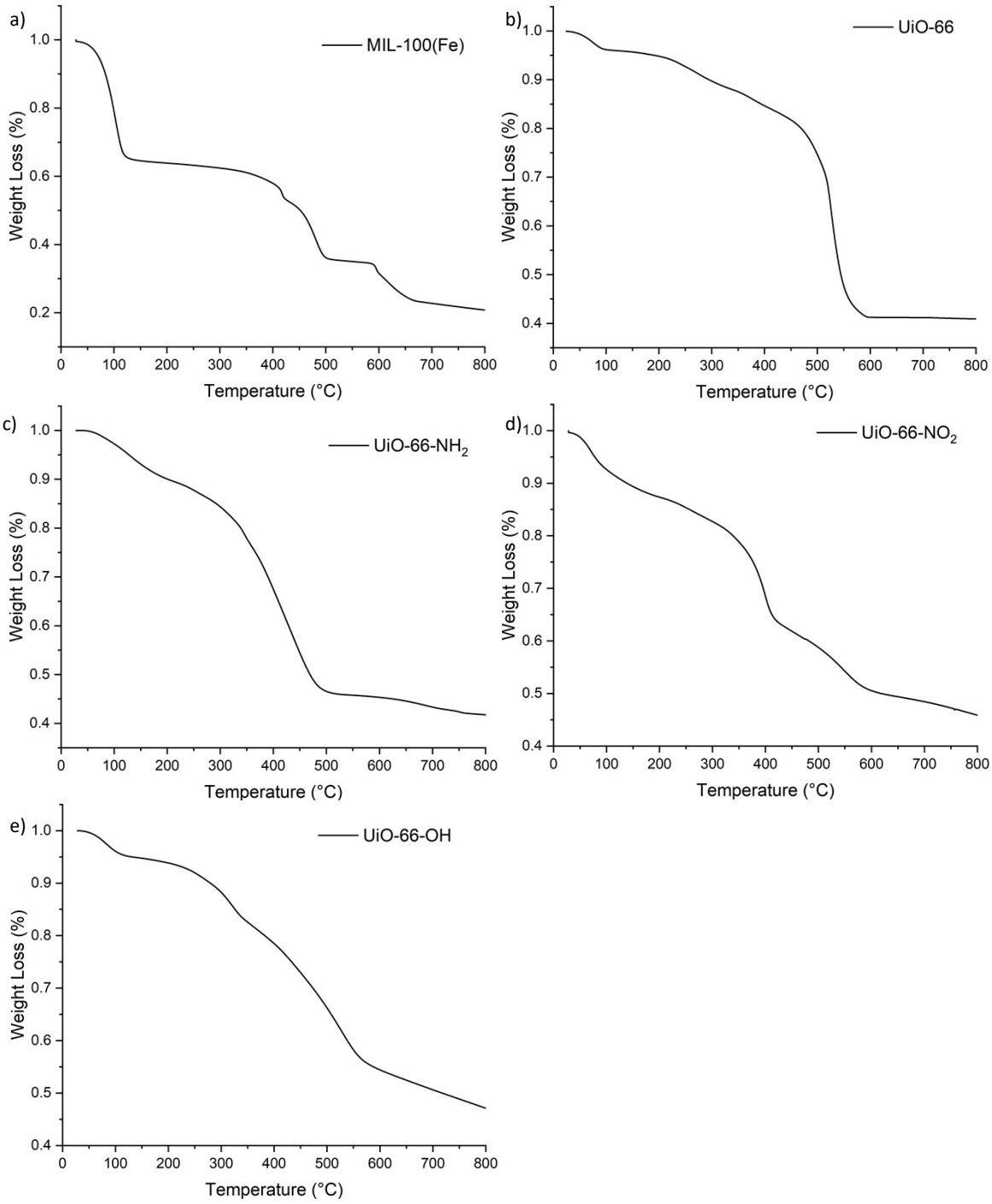


Fig. S4: TGA curves from synthesized MOFs at a ramp rate of 20 °C per minute: (a) MIL-100(Fe), (b) UiO-66, (c) UiO-66-NH2, (d) UiO-66-NO2, and (e) UiO-66-OH.

Particle Size Analysis:

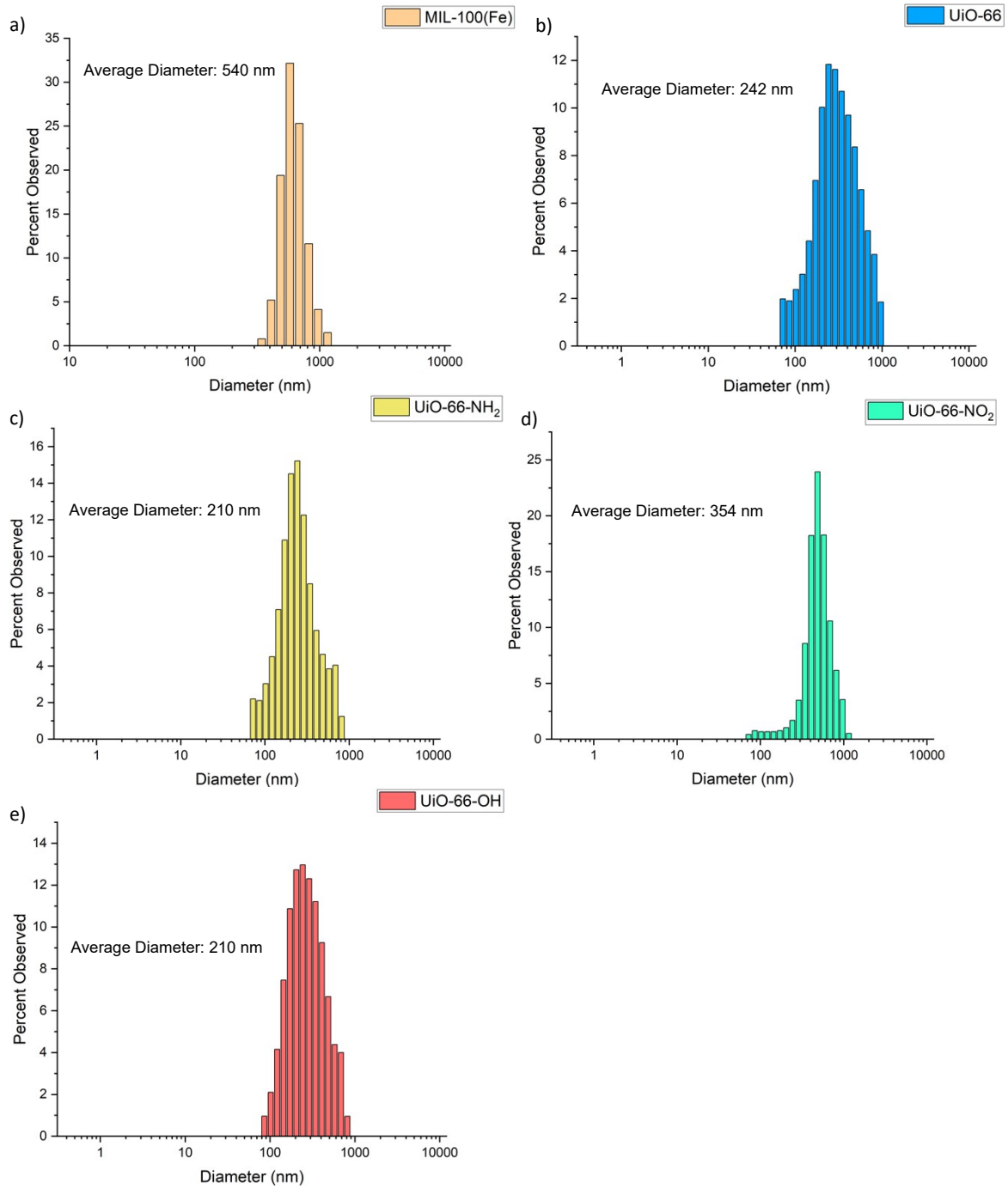


Fig. S5: Particle size analysis results after size exclusion protocol from synthesized MOFs (a) MIL-100(Fe), (b) UiO-66, (c) UiO-66-NH2, (d) UiO-66-NO2, and (e) UiO-66-OH.

Scanning Electron Microscope Imaging:

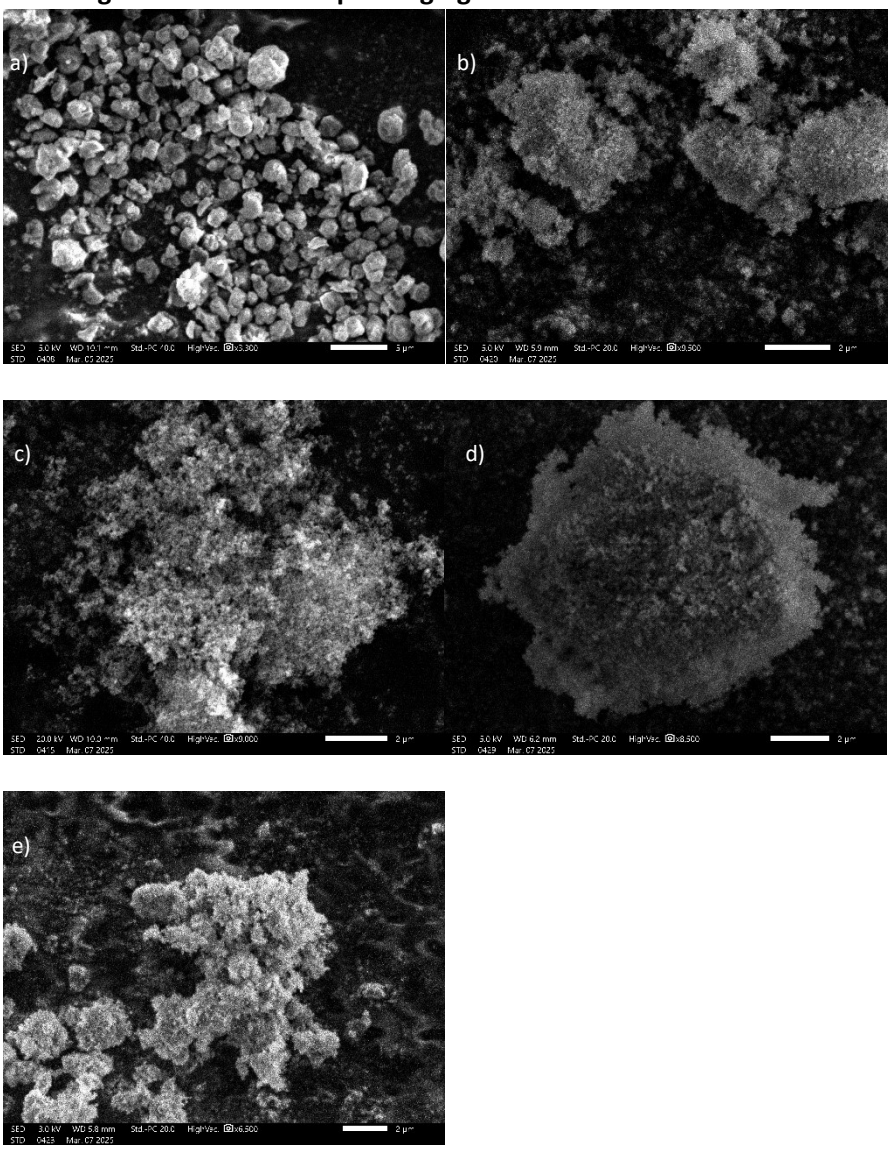


Fig. S6: SEM images of as synthesized MOFs to observe the morphology of a) MIL-100, b) UiO-66, c) UiO-66-NH_2, d) UiO-66-NO_2, and e) UiO-66-OH.

Zeta Potential Analysis:

Table S1: Observed Zeta Potential of synthesized MOFs

MOF	Zeta Potential (mV)
MIL-100(Fe)	-12.6
UiO-66	-7.60
UiO-66-NH2	14.7
UiO-66-NO2	-11.5
UiO-66-OH	-17.3

Completed Release Data:

Table S2: Final concentration in solution and percent release of dye from MOFs during testing

Trial	Condition	MO		MB	
		µg/mg MOF	Percent Released	µg/mg MOF	Percent Released
Buffer Concentration	0 mM	0.12	9.57%	0.03	2.56%
	20 mM	0.14	11.3%	0.24	18.1%
	50 mM	0.16	12.6%	0.47	35.0%
	100 mM	0.19	15.1%	0.16	12.0%
Loading Concentration	Light Loading	0.14	8.21%	0.24	18.1%
	Standard Loading	0.25	25.6%	0.18	17.8%
Functional Groups	UiO-66	0.53	65.9%	0.19	92.6%
	UiO-66-NH_2	0.26	36.6%	0.09	87.9%
	UiO-66-NO_2	.16	22.1%	.14	69.8%
	UiO-66-OH	0.25	48.4%	0.04	14.9%
Polyelectrolyte Addition	Standard	0.25	25.6%	0.18	17.8%
	0.1% HEP	0.50	47.7%	0.07	0.08%
	0.1% PEI	0.43	40.6%	0.23	25.5%
	1.0% PEI	0.47	97.9%	1.01	~100%

Fitted Data

Buffer Concentration Testing:

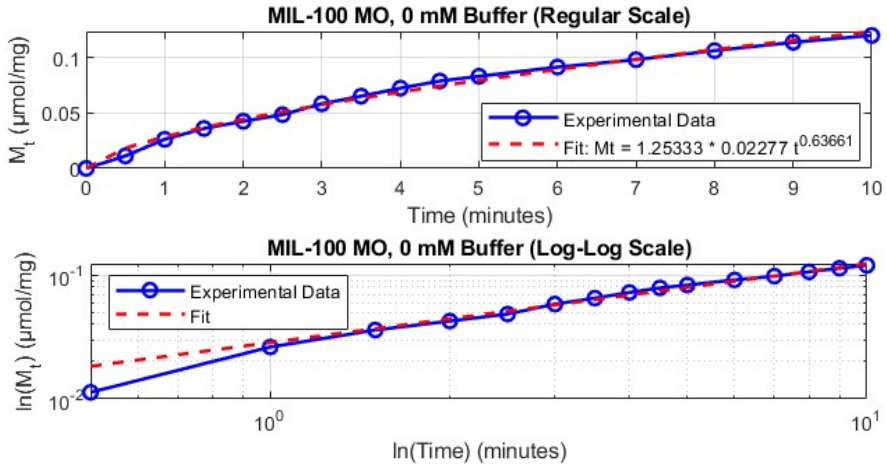


Fig. S7: Plots and fitted curves of MO release with 0 mM buffer added using Eq. 1 (R^2 = 0.99).

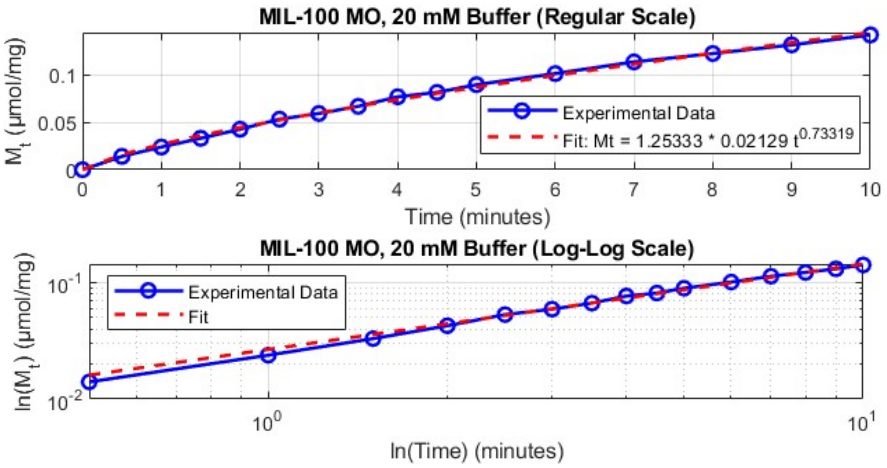


Fig. S8: Plots and fitted curves of MO release with 20 mM buffer added using Eq. 1 (R^2 = 0.99).

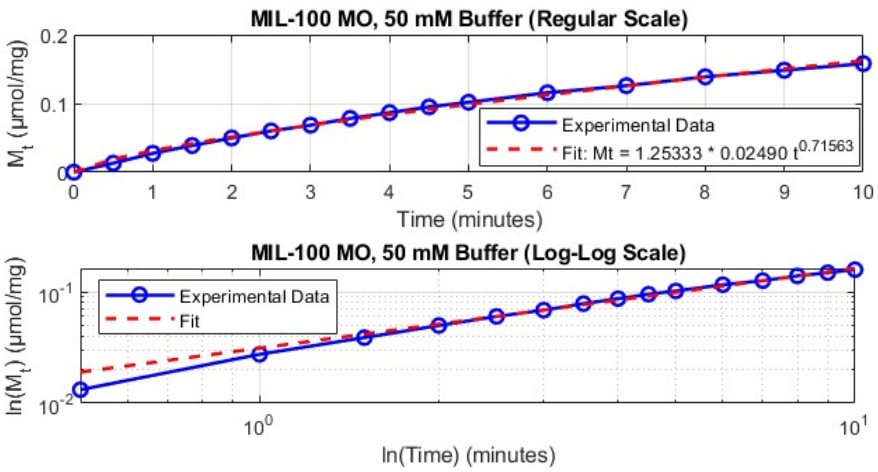


Fig. S9: Plots and fitted curves of MO release from MIL-100 with 50 mM buffer added using Eq. 1 (R^2 = 0.99).

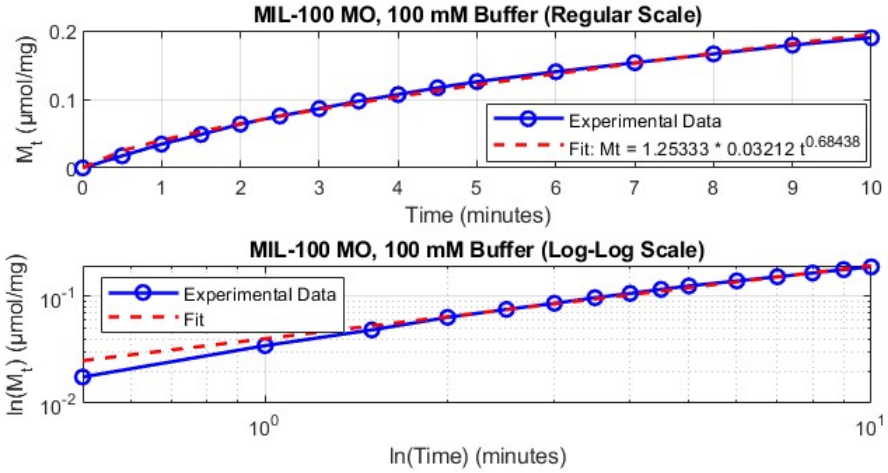


Fig. S10: Plots and fitted curves of MO release from MIL-100 with 100 mM buffer added using Eq. 1 (R^2 = 0.99).

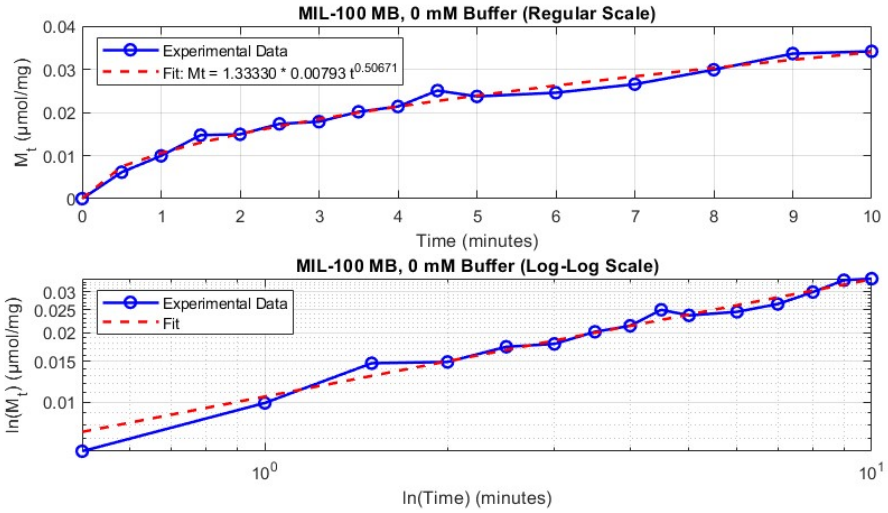


Fig. S11: Plots and fitted curves of MB release from MIL-100 with 0 mM buffer added using Eq. 1 (R^2 = 0.99).

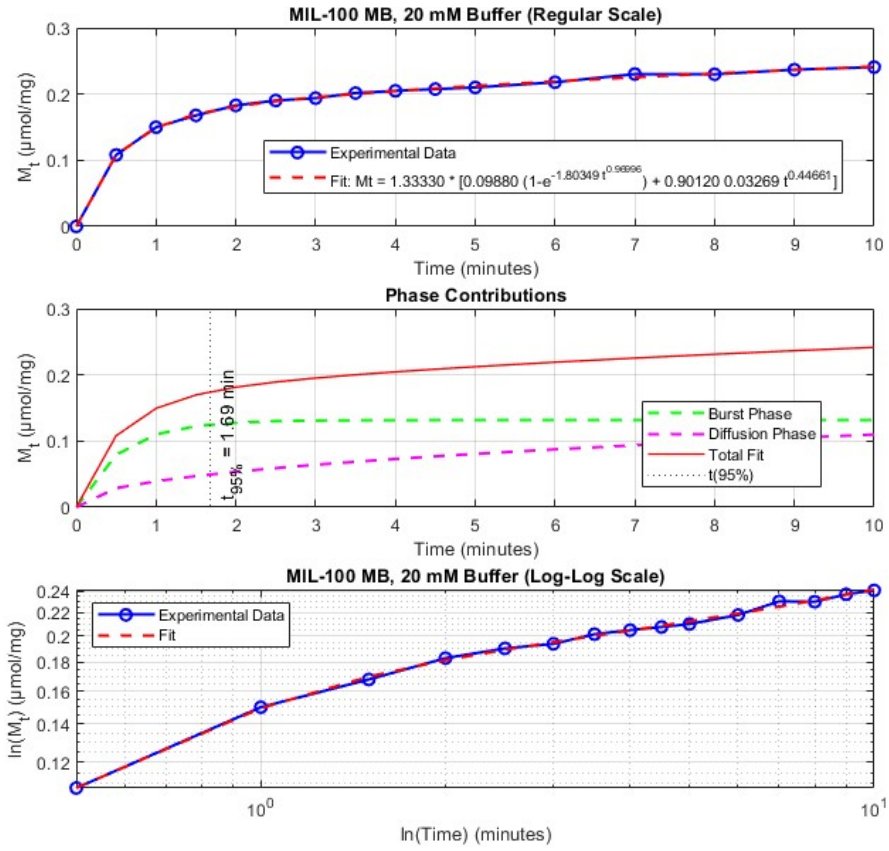


Fig. S12: Plots and fitted curves of MB release from MIL-100 with 20 mM buffer added using Eq. 2 (R^2 = 0.99).

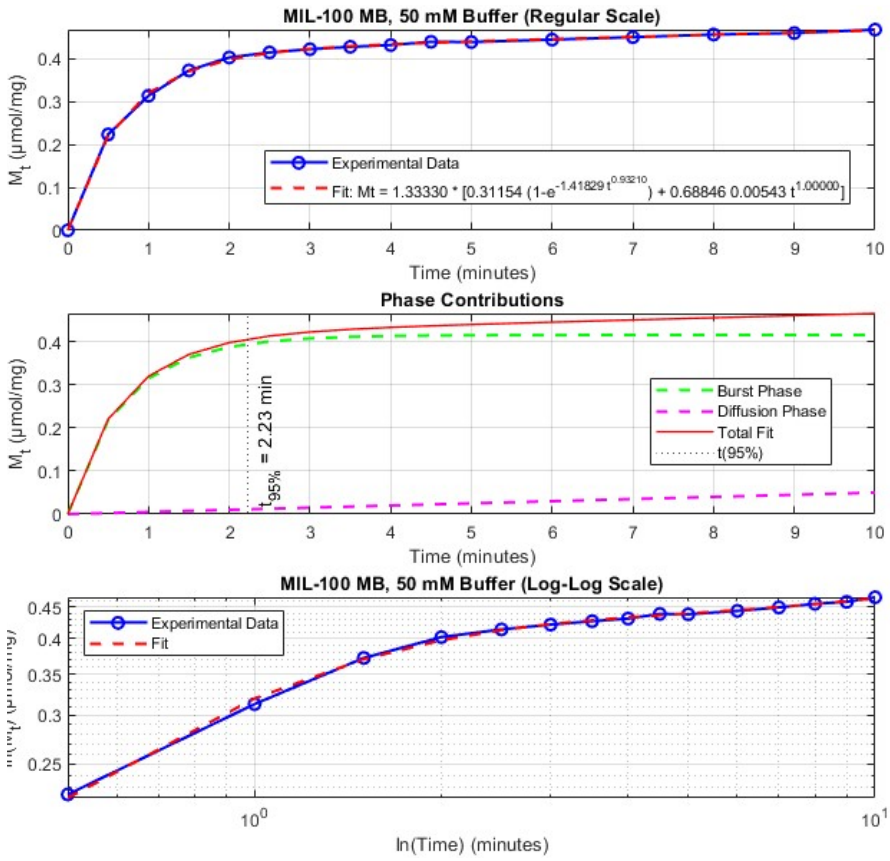


Fig. S13: Plots and fitted curves of MB release from MIL-100 with 50 mM buffer added using Eq. 2 (R^2 = 0.99).

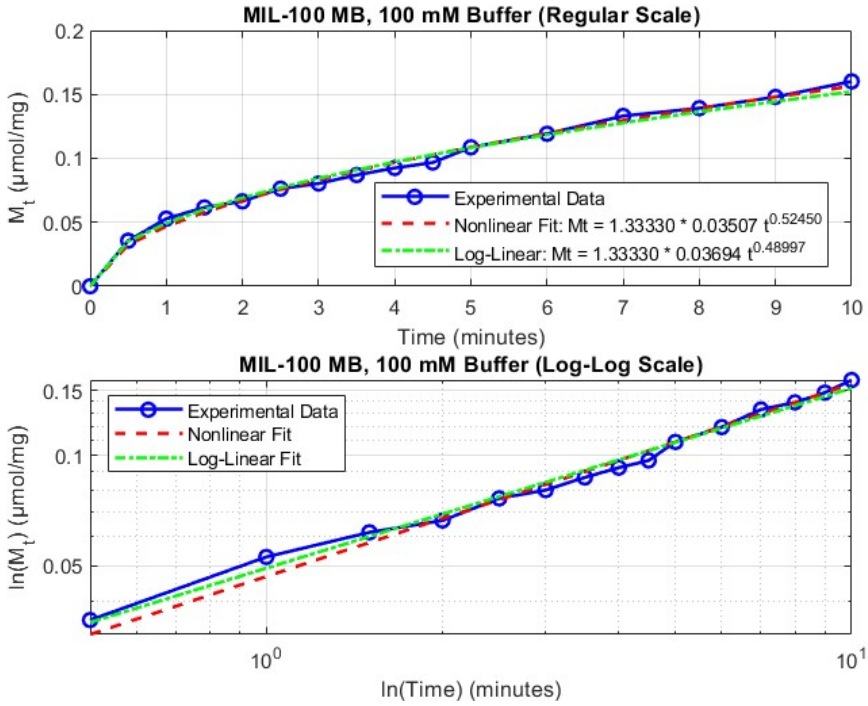


Fig. S14: Plots and fitted curves of MB release from MIL-100 with 100 mM buffer added using Eq. 1 (R^2 = 0.99).

Functional Group Testing:

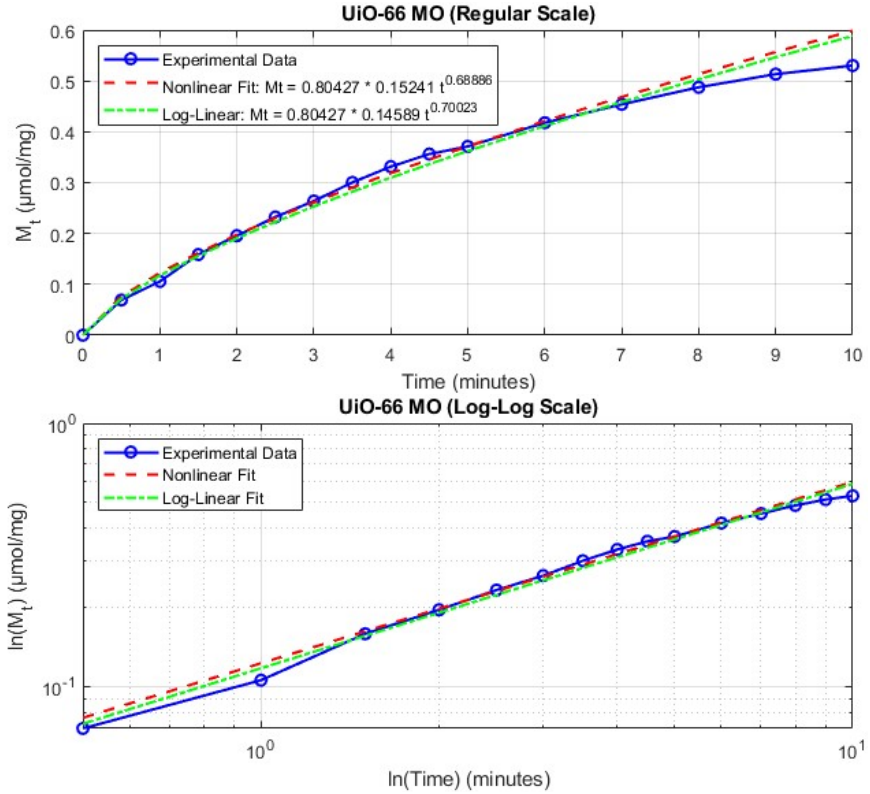
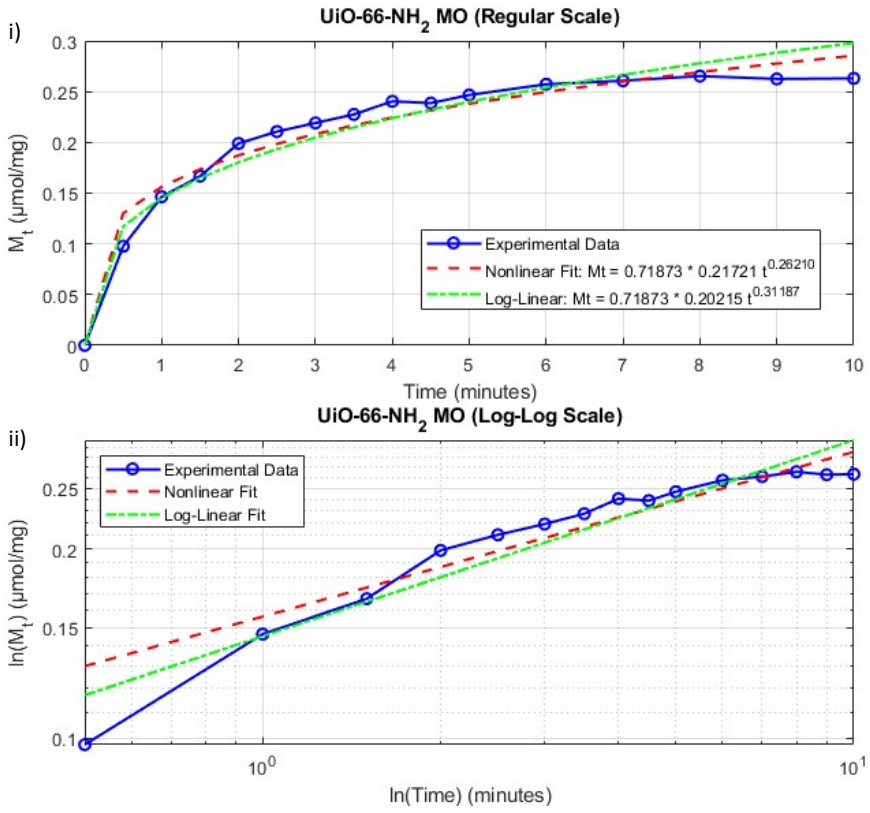


Fig. S15: Plots and fitted curves of MO release from UiO-66 using Eq. 1 (R^2 = 0.99)



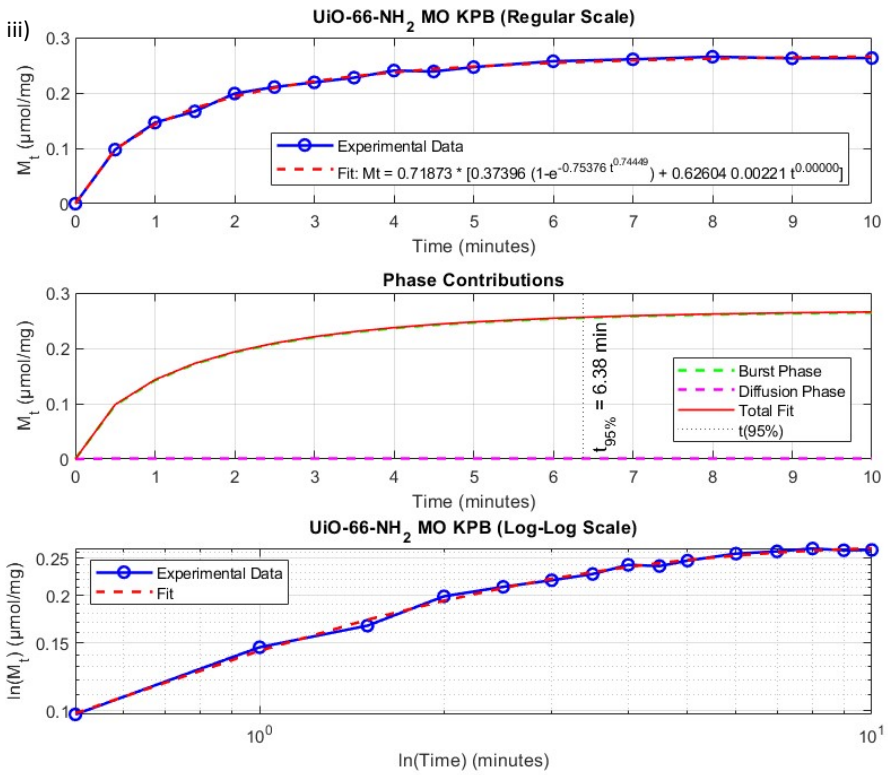
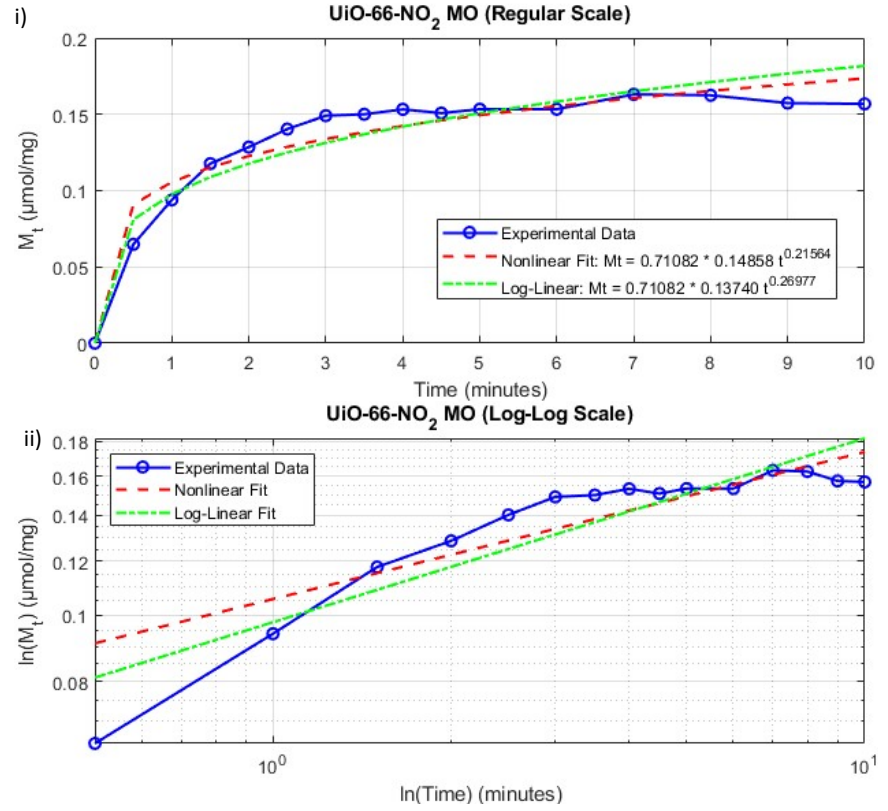


Fig. S16: Plots and fitted curves of MB release from UiO-66 using Eq. 1 (I and ii, R^2 = 0.91) and Eq. 2 (iii, R^2 = 0.99)



iii)

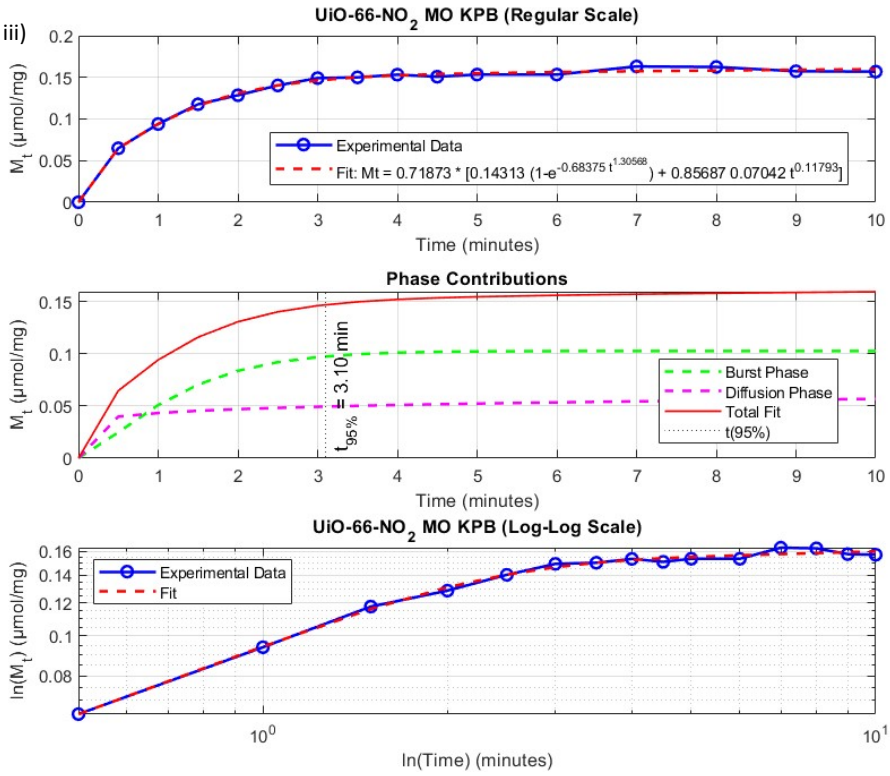


Fig. S17: Plots and fitted curves of MB release from UiO-66-NO_2 using Eq. 1 (I and ii, R^2 = 0.82) and Eq. 2 (iii, R^2 = 0.99)

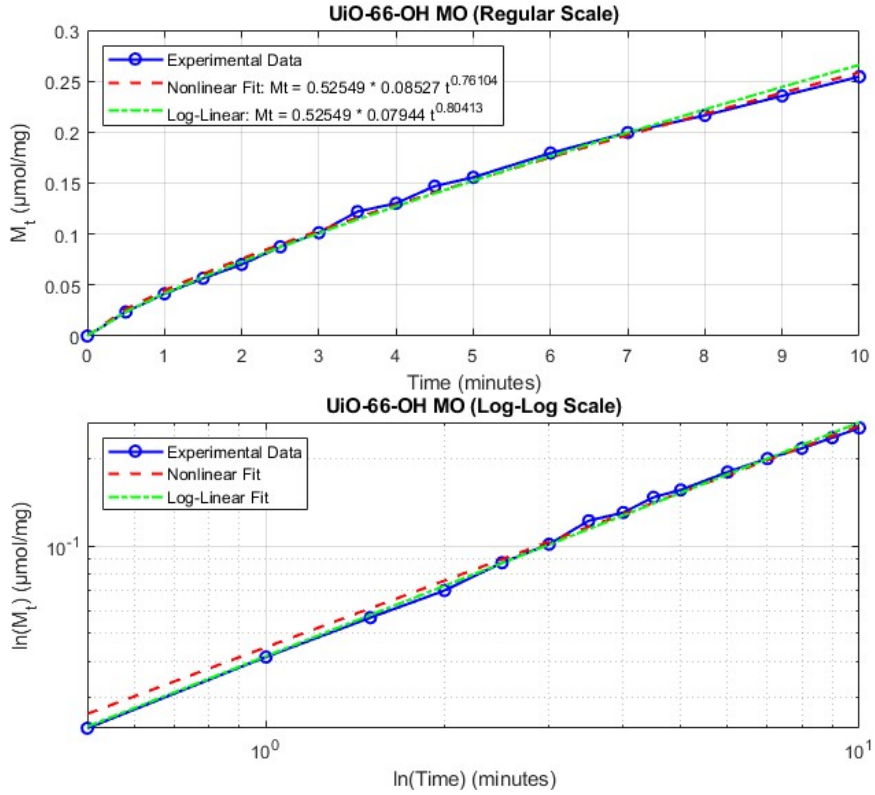


Fig. S18: Plots and fitted curves of MB release from UiO-66-NO_2 using Eq. 1 (R^2 = 0.99).

i)

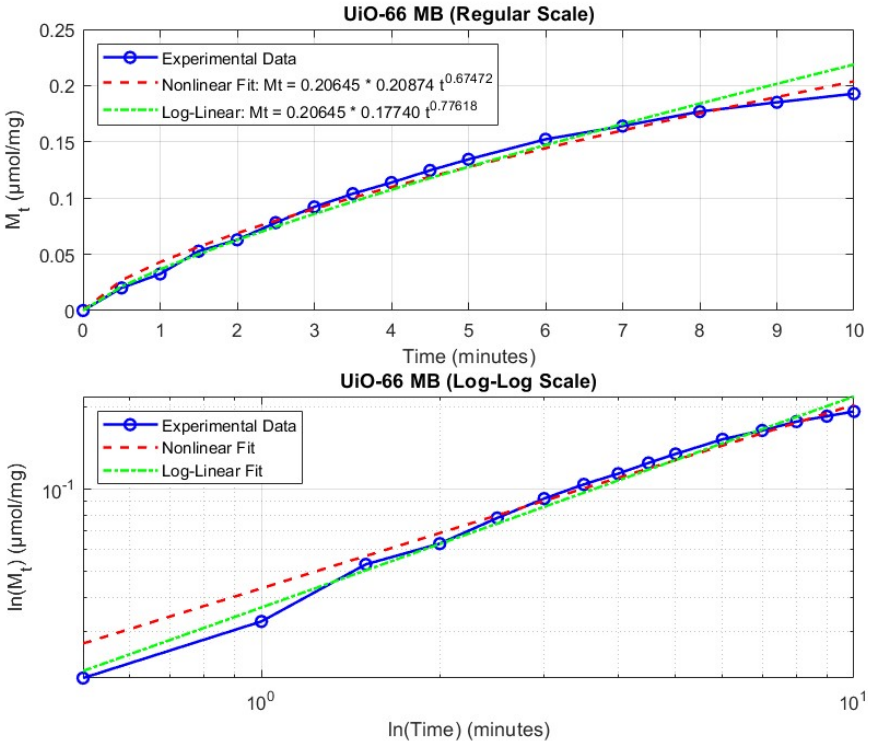


Fig. S19: Plots and fitted curves of MB release from UiO-66 using Eq. 1 (R^2 = 0.99)

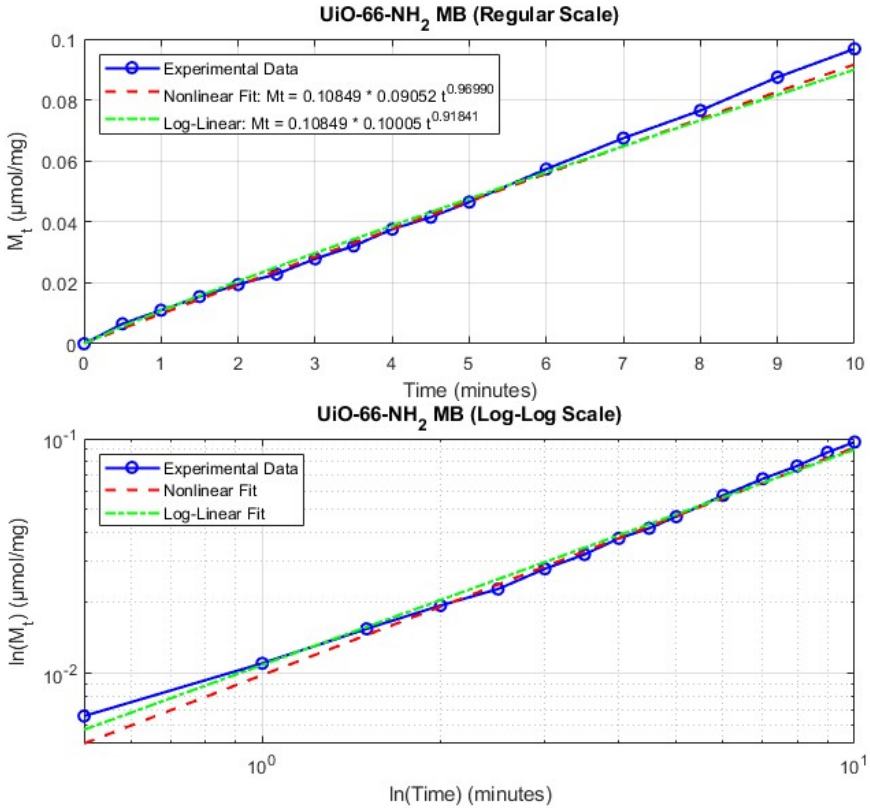


Fig. S20: Plots and fitted curves of MB release from UiO-66-NH_2 using Eq. 1 (R^2 = 0.99)

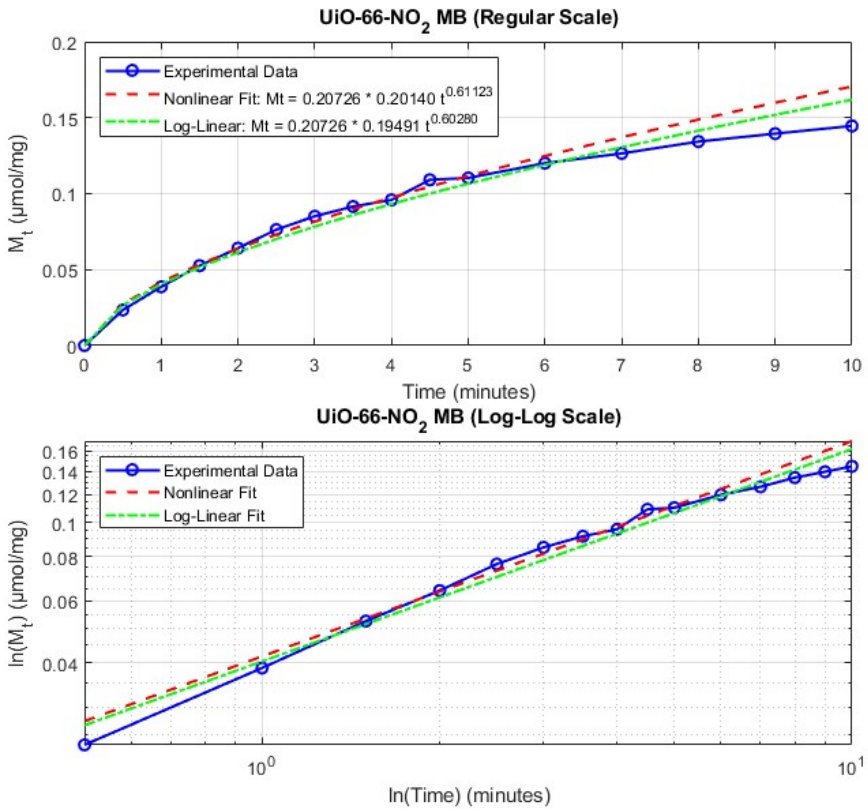


Fig. S21: Plots and fitted curves of MB release from UiO-66-NO_2 using Eq. 1 (R^2 = 0.98).

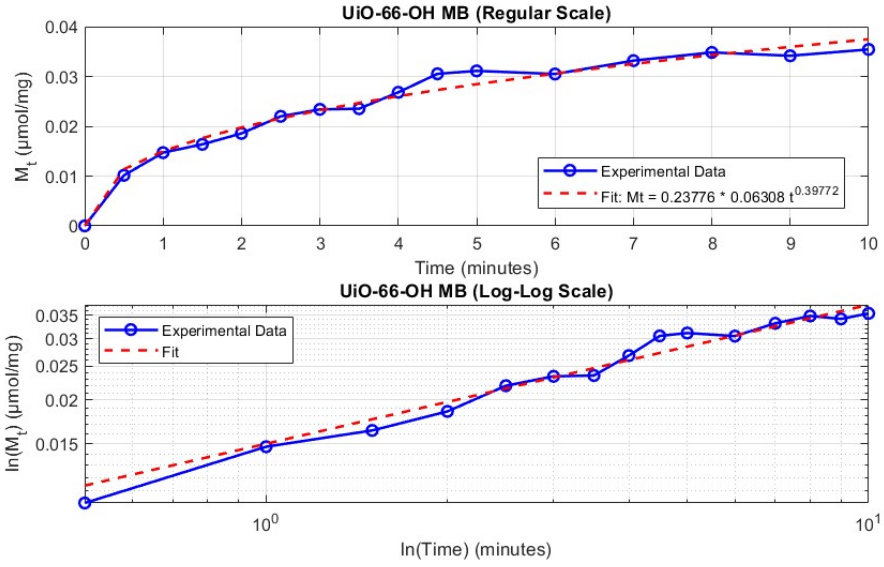


Fig. S22: Plots and fitted curves of MB release from UiO-66-OH using Eq. 1 (R^2 = 0.96).

Polyelectrolyte Testing:

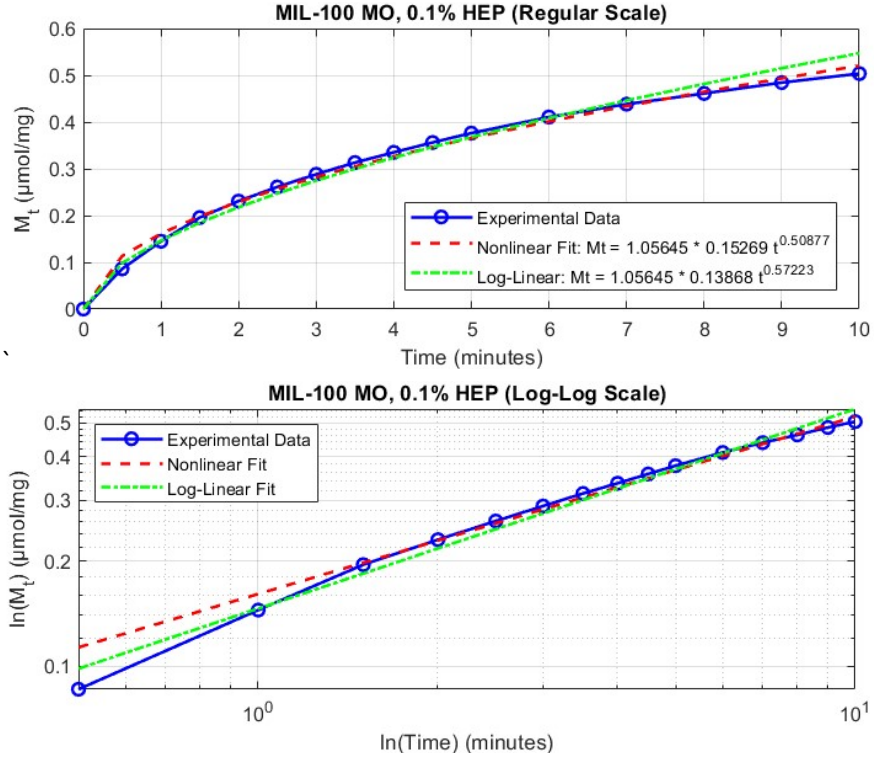


Fig. S23: Plots and fitted curves of MO release from MIL-100 in the presence of 0.1% Heparin using Eq. 1 (R^2 = 0.99).

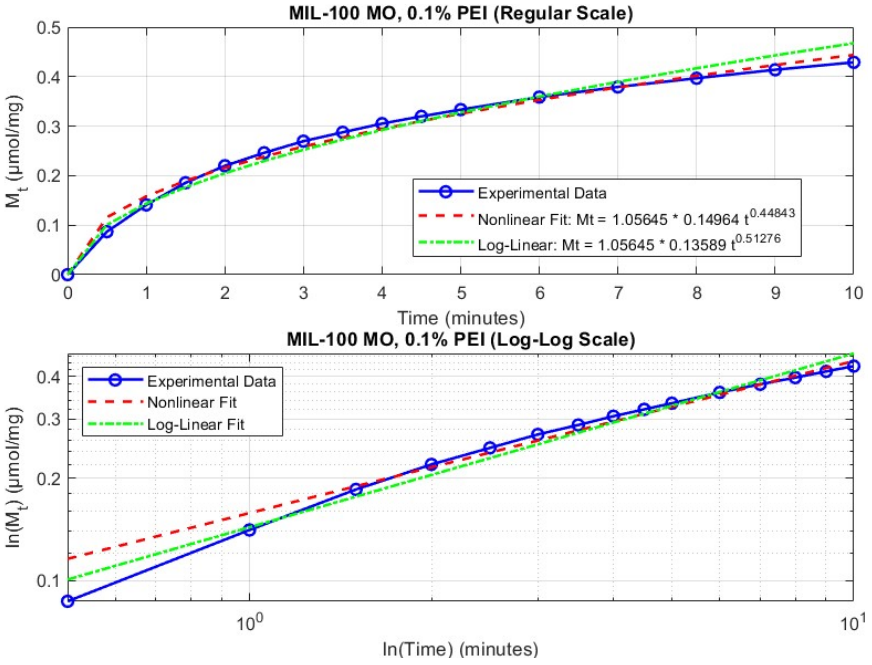


Fig. S24: Plots and fitted curves of MO release from MIL-100 in the presence of 0.1% PEI using Eq. 1 (R^2 = 0.99).

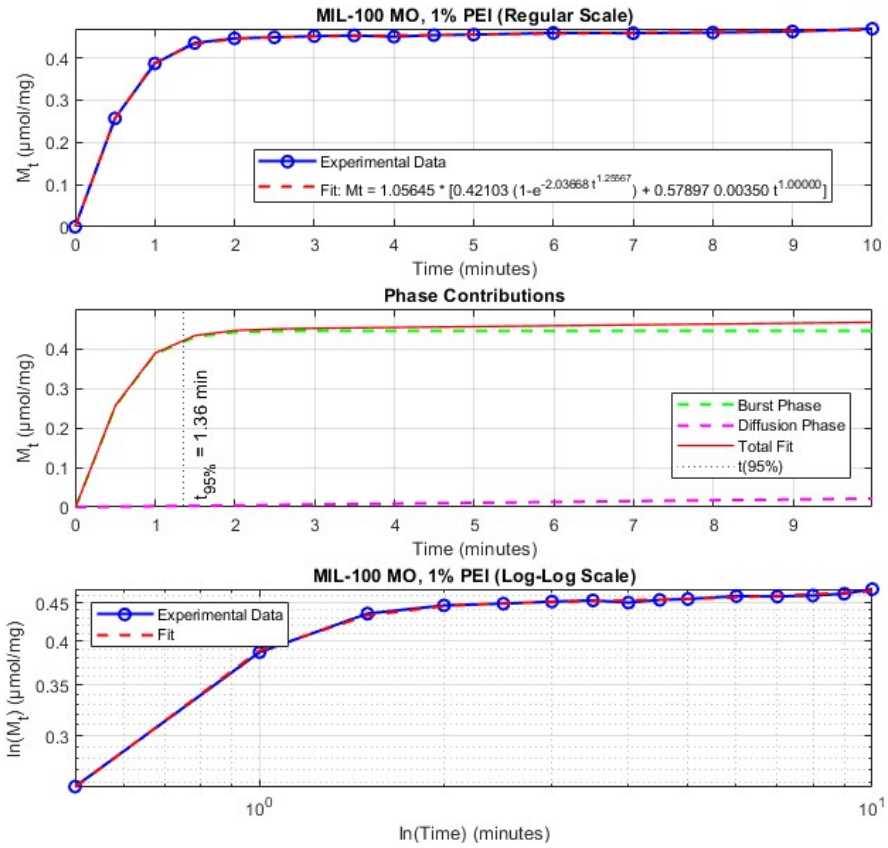


Fig. S25: Plots and fitted curves of MO release from MIL-100 in the presence of 1% PEI using Eq. 2 (R^2 = 0.99).

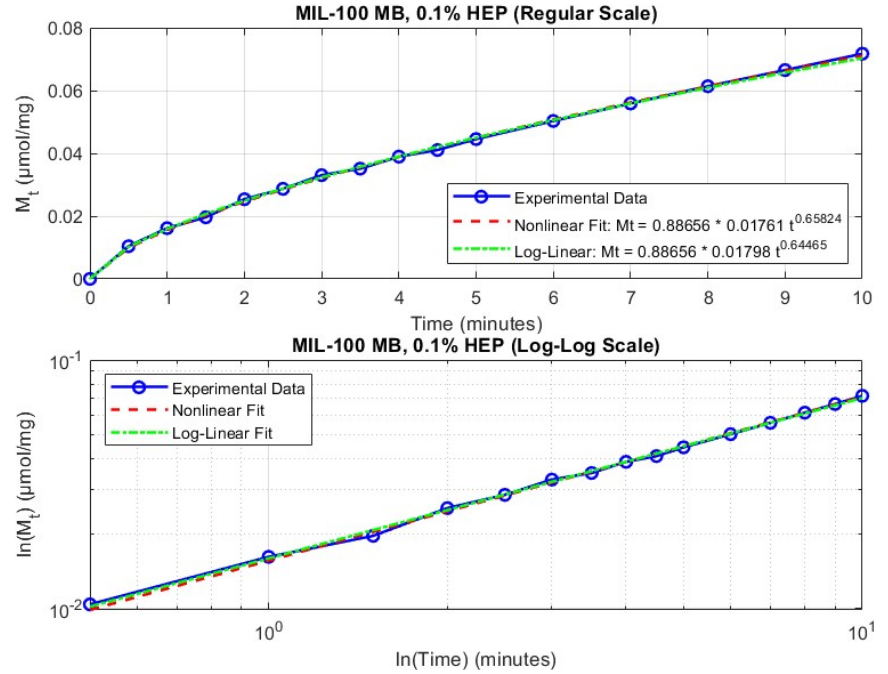


Fig. S26: Plots and fitted curves of MB release from MIL-100 in the presence of 0.1% HEP using Eq. 1 (R^2 = 0.99).

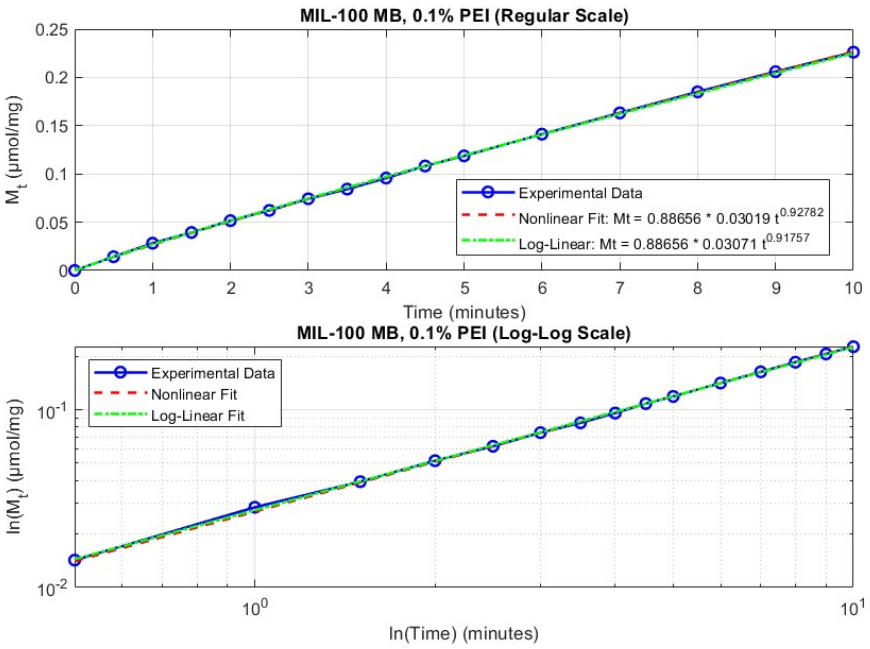


Fig. S27: Plots and fitted curves of MB release from MIL-100 in the presence of 0.1% PEI using Eq. 1 (R^2 = 0.99).

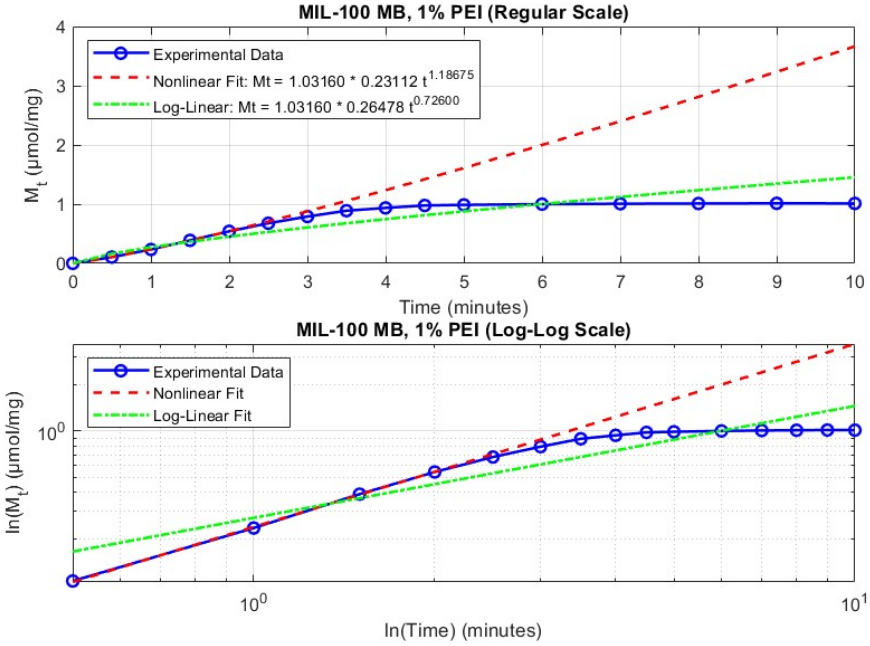


Fig. S28: Plots and fitted curves of MB release from MIL-100 in the presence of 1% PEI using Eq. 1 (R^2 = 0.99). It should be noted that, unlike other trials, this trial resulted in one release type until fully released which is why the Korsmeyer-Peppas equation was selected for use.

MATLAB Scripting:

```
% Close all Fig.s and clear command window for a clean start

close all;
```

```
clc;

% Fixed time points
time = [0, 0.5, 1, 1.5, 2, 2.5, 3, 3.5, 4, 4.5, 5, 6, 7, 8, 9, 10]; % Time in minutes %You must adjust these X variables to fit your own plots.

% File to store M_infinity values
minf_file = 'Minfinity_storage.mat';

% Check if storage file exists and load it, or initialize an empty struct
if exist(minf_file, 'file')
    load(minf_file, 'minf_data');
else
    minf_data = struct(); % Empty struct to store M_infinity values
end

% Model selection
disp('Select the model to fit:');
disp('1. Korsmeyer-Peppas Model (Mt = M_infinity * k * t^n)');
disp('2. Two-Phase Model (Mt = M_infinity * [A * (1 - exp(-k1 * t^n1)) + B * k2 * t^n2])');
model_choice = input('Enter your choice (1 or 2): ');

% Manage M_infinity input with storage, recall, or optimization options
disp('M_infinity Management:');
disp('1. Enter a new M_infinity value');
disp('2. Recall a stored M_infinity value');
disp('3. Optimize M_infinity during fitting');
choice = input('Enter your choice (1, 2, or 3): ');

if choice == 1
    disp('Enter the value for M_infinity (e.g., 4.622):');
    M_infinity = input('M_infinity = ');
    label = input('Enter a label for this M_infinity (e.g., Run1): ', 's');
    minf_data.(label) = M_infinity;
    save(minf_file, 'minf_data');
    fprintf('Stored M_infinity = %.5f under label "%s"\n', M_infinity, label);
elseif choice == 2
    if isempty(fieldnames(minf_data))
        error('No stored M_infinity values available. Please enter a new one first.');
    end
    disp('Stored M_infinity values:');
    labels = fieldnames(minf_data);
    for i = 1:length(labels)
        fprintf('%d. %s: %.5f\n', i, labels{i}, minf_data.(labels{i}));
    end
    selection = input('Enter the number of the M_infinity to recall: ');
    if selection < 1 || selection > length(labels)
        error('Invalid selection. Please choose a number between 1 and %d.', length(labels));
    end
    M_infinity = minf_data.(labels{selection});
    fprintf('Recalled M_infinity = %.5f from label "%s"\n', M_infinity, labels{selection});
elseif choice == 3
    M_infinity = NaN; % Placeholder; will be optimized
```

```
    fprintf('M_infinity will be optimized during the nonlinear fit.\n');
else
    error('Invalid choice. Please enter 1, 2, or 3.');
end

% Prompt user to input all 16 Mt values at once
disp('Enter all 16 Mt values separated by spaces or commas (e.g., 0 0.045263983 ...):');
Mt_input = input('Mt = ', 's');
Mt = str2num(Mt_input);

% Validate input
if length(Mt) ~= 16
    error('Error: You must enter exactly 16 Mt values. You provided %d values.', length(Mt));
end

% Prompt user for a custom chart title
chart_title = input('Enter a title for the charts: ', 's');

% Option to fit only early data (Mt/M_infinity < 0.6)
fit_early = input('Fit only early data (Mt/M_infinity < 0.6)? (1 = Yes, 0 = No): ');

% Define model functions and handle A/B
if model_choice == 1
    % Korsmeyer-Peppas Model
    if choice == 3
        korsmeyer_peppas = @(params, t) params(1) * params(2) * t.^params(3); % [M_infinity, k, n]
        initial_params = [max(Mt)*1.1, 0.01, 0.7];
    else
        korsmeyer_peppas = @(params, t) M_infinity * params(1) * t.^params(2); % [k, n]
        initial_params = [0.01, 0.7];
    end
elseif model_choice == 2
    % Two-Phase Model: Mt = M_infinity * [A * (1 - exp(-k1 * t^n1)) + B * k2 * t^n2]
    disp('For Two-Phase Model:');
    disp('1. Optimize A and B (default)');
    disp('2. Manually set A (B = 1 - A)');
    ab_choice = input('Enter your choice (1 or 2): ');

    if ab_choice == 1
        % Optimize A
        if choice == 3
            two_phase = @(params, t) params(1) * (params(2) * (1 - exp(-params(3) * t.^params(4))) + (1-params(2)) *
params(5) * t.^params(6));
            initial_params = [max(Mt)*1.1, 0.8, 1, 1, 0.01, 0.5]; % [M_infinity, A, k1, n1, k2, n2]
        else
            two_phase = @(params, t) M_infinity * (params(1) * (1 - exp(-params(2) * t.^params(3))) + (1-params(1)) *
params(4) * t.^params(5));
            initial_params = [1, 0.8, 1, 0.01, 0.5]; % [A, k1, n1, k2, n2]
        end
    elseif ab_choice == 2
        % Manually set A
        A_fixed = input('Enter value for A (0 to 1): ');
        if A_fixed < 0 || A_fixed > 1
```

```
            error('A must be between 0 and 1.');
        end
        B_fixed = 1 - A_fixed;
        fprintf('Using fixed A = %.5f, B = %.5f\n', A_fixed, B_fixed);
        if choice == 3
            two_phase = @(params, t) params(1) * (A_fixed * (1 - exp(-params(2) * t.^params(3))) + B_fixed * params(4)
* t.^params(5));
            initial_params = [max(Mt)*1.1, 1, 1, 0.01, 0.5]; % [M_infinity, k1, n1, k2, n2] This script does a good job at
fitting the data, but you may need to adjust the "A" term to get a better fit.
        else
            two_phase = @(params, t) M_infinity * (A_fixed * (1 - exp(-params(1) * t.^params(2))) + B_fixed * params(3)
* t.^params(4));
            initial_params = [1, 1, 0.01, 0.5]; % [k1, n1, k2, n2]
        end
    else
        error('Invalid choice. Please enter 1 or 2.');
    end
else
    error('Invalid model choice. Please enter 1 or 2.');
end

% Nonlinear least-squares fit (excluding t=0)
t_fit = time(2:end);
Mt_fit = Mt(2:end);

% Apply early data filter if selected and M_infinity is provided
if fit_early && choice ~= 3
    early_idx = Mt_fit < 0.6 * M_infinity;
    t_fit = t_fit(early_idx);
    Mt_fit = Mt_fit(early_idx);
    fprintf('Fitting only early data: %d points where Mt/M_infinity < 0.6\n', sum(early_idx));
end

% Perform fitting
options = optimoptions('lsqcurvefit', 'MaxIterations', 1000, 'TolFun', 1e-8, 'Display', 'off');
if model_choice == 1
    [params_opt, resnorm] = lsqcurvefit(korsmeyer_peppas, initial_params, t_fit, Mt_fit, [], [], options);
    if choice == 3
        M_infinity_opt = params_opt(1);
        k_opt = params_opt(2);
        n_opt = params_opt(3);
        Mt_predicted = korsmeyer_peppas(params_opt, time);
    else
        k_opt = params_opt(1);
        n_opt = params_opt(2);
        M_infinity_opt = M_infinity;
        Mt_predicted = korsmeyer_peppas(params_opt, time);
    end
else % model_choice == 2
    % Define bounds based on number of parameters
    if ab_choice == 1
        if choice == 3
            lb = [0, 0, 0, 0, 0, 0]; % [M_infinity, A, k1, n1, k2, n2]
```

```
        ub = [Inf, 1, Inf, 2, Inf, 1];
    else
        lb = [0, 0, 0, 0, 0]; % [A, k1, n1, k2, n2]
        ub = [1, Inf, 2, Inf, 1];
    end
else % ab_choice == 2
    if choice == 3
        lb = [0, 0, 0, 0, 0]; % [M_infinity, k1, n1, k2, n2]
        ub = [Inf, Inf, 2, Inf, 1];
    else
        lb = [0, 0, 0, 0]; % [k1, n1, k2, n2]
        ub = [Inf, 2, Inf, 1];
    end
end
[params_opt, resnorm] = lsqcurvefit(two_phase, initial_params, t_fit, Mt_fit, lb, ub, options);
if ab_choice == 1
    if choice == 3
        M_infinity_opt = params_opt(1);
        A_opt = params_opt(2);
        k1_opt = params_opt(3);
        n1_opt = params_opt(4);
        k2_opt = params_opt(5);
        n2_opt = params_opt(6);
        B_opt = 1 - A_opt;
        Mt_predicted = two_phase(params_opt, time);
    else
        A_opt = params_opt(1);
        k1_opt = params_opt(2);
        n1_opt = params_opt(3);
        k2_opt = params_opt(4);
        n2_opt = params_opt(5);
        B_opt = 1 - A_opt;
        M_infinity_opt = M_infinity;
        Mt_predicted = two_phase(params_opt, time);
    end
else % ab_choice == 2
    if choice == 3
        M_infinity_opt = params_opt(1);
        k1_opt = params_opt(2);
        n1_opt = params_opt(3);
        k2_opt = params_opt(4);
        n2_opt = params_opt(5);
        A_opt = A_fixed;
        B_opt = B_fixed;
        Mt_predicted = two_phase(params_opt, time);
    else
        k1_opt = params_opt(1);
        n1_opt = params_opt(2);
        k2_opt = params_opt(3);
        n2_opt = params_opt(4);
        A_opt = A_fixed;
        B_opt = B_fixed;
        M_infinity_opt = M_infinity;
```

```
            Mt_predicted = two_phase(params_opt, time);
        end
    end
end

% Calculate R-squared
if fit_early && choice ~= 3
    Mt_fit_full = Mt(2:end);
    ss_tot = sum((Mt_fit_full(early_idx) - mean(Mt_fit_full(early_idx))).^2);
    ss_res = resnorm;
else
    ss_tot = sum((Mt_fit - mean(Mt_fit)).^2);
    ss_res = resnorm;
end
r_squared = 1 - (ss_res / ss_tot);

% Display results
fprintf('\nFit Results:\n');
fprintf('M_infinity = %.5f (umol/mg)\n', M_infinity_opt);
if model_choice == 1
    fprintf('k = %.5f (normalized units)\n', k_opt);
    fprintf('n = %.5f\n', n_opt);
else
    fprintf('A = %.5f (fraction of burst phase)\n', A_opt);
    fprintf('k1 = %.5f (burst rate constant)\n', k1_opt);
    fprintf('n1 = %.5f (burst exponent)\n', n1_opt);
    fprintf('B = %.5f (fraction of diffusion phase)\n', B_opt);
    fprintf('k2 = %.5f (diffusion rate constant)\n', k2_opt);
    fprintf('n2 = %.5f (diffusion exponent)\n', n2_opt);
end
fprintf('R-squared = %.5f\n', r_squared);
fprintf('Residual norm = %.5e\n', resnorm);

% Plotting
Fig.('Name', 'Model Fit', 'NumberTitle', 'off');
subplot(3,1,1); % Regular scale with total fit
plot(time, Mt, 'bo-', 'LineWidth', 1.5, 'DisplayName', 'Experimental Data');
hold on;
if model_choice == 1
    plot(time, Mt_predicted, 'r--', 'LineWidth', 1.5, ...
        'DisplayName', sprintf('Fit: Mt = %.5f * %.5f t^{%.5f}', M_infinity_opt, k_opt, n_opt));
else
    plot(time, Mt_predicted, 'r--', 'LineWidth', 1.5, ...
        'DisplayName', sprintf('Fit: Mt = %.5f * [%.5f (1-e^{-%.5f t^{%.5f}}) + %.5f %.5f t^{%.5f}]', ...
        M_infinity_opt, A_opt, k1_opt, n1_opt, B_opt, k2_opt, n2_opt));
end
xlabel('Time (minutes)');
ylabel('M_t (µmol/mg)');
title([chart_title ' (Regular Scale)']);
legend('Location', 'NorthWest');
grid on;
hold off;
```

```
subplot(3,1,2); % Separate contributions
if model_choice == 2
   M_burst = M_infinity_opt * A_opt * (1 - exp(-k1_opt * time.^n1_opt));
   M_diffusion = M_infinity_opt * B_opt * k2_opt * time.^n2_opt;
   rate_burst = M_infinity_opt * A_opt * k1_opt * n1_opt * time.^(n1_opt - 1) .* exp(-k1_opt * time.^n1_opt);
   rate_diffusion = M_infinity_opt * B_opt * k2_opt * n2_opt * time.^(n2_opt - 1);
   plot(time, M_burst, 'g--', 'LineWidth', 1.5, 'DisplayName', 'Burst Phase');
   hold on;
   plot(time, M_diffusion, 'm--', 'LineWidth', 1.5, 'DisplayName', 'Diffusion Phase');
   plot(time, Mt_predicted, 'r-', 'LineWidth', 1, 'DisplayName', 'Total Fit');
   % Estimate transition time (95% of burst)
   t_transition = (2.995 / k1_opt)^(1/n1_opt);
   xline(t_transition, 'k:', 'Label', sprintf('t_{95%%} = %.2f min', t_transition), ...
       'LabelVerticalAlignment', 'bottom', 'LabelHorizontalAlignment', 'right');
   xlabel('Time (minutes)');
   ylabel('M_t (µmol/mg)');
   title('Phase Contributions');
   legend('Location', 'NorthWest');
   grid on;
   hold off;

   % Display phase contributions and rates data
   fprintf('\nPhase Contributions and Rates Data (Time, M_burst, M_diffusion, Rate_burst, Rate_diffusion):\n');
   fprintf('Time (min) | M_burst (umol/mg) | M_diffusion (umol/mg) | Rate_burst (umol/mg/min) | Rate_diffusion
(umol/mg/min)\n');
   fprintf('----------------------------------------------------------------------------------------\n');
   for i = 1:length(time)
      fprintf('%.2f      | %.5f         | %.5f           | %.5f           | %.5f\n', ...
          time(i), M_burst(i), M_diffusion(i), rate_burst(i), rate_diffusion(i));
   end
end

subplot(3,1,3); % Log-log scale
valid_idx = time > 0;
loglog(time(valid_idx), Mt(valid_idx), 'bo-', 'LineWidth', 1.5, 'DisplayName', 'Experimental Data');
hold on;
loglog(time(valid_idx), Mt_predicted(valid_idx), 'r--', 'LineWidth', 1.5, 'DisplayName', 'Fit');
xlabel('ln(Time) (minutes)');
ylabel('ln(M_t) (µmol/mg)');
title([chart_title ' (Log-Log Scale)']);
legend('Location', 'NorthWest');
grid on;
hold off;

% Residuals plot
Fig.('Name', 'Residuals Analysis', 'NumberTitle', 'off');
plot(time, Mt - Mt_predicted, 'ko-', 'LineWidth', 1.5);
xlabel('Time (minutes)');
ylabel('Residuals (µmol/mg)');
title(['Residuals of ' chart_title]);
grid on;

% Display transition time
```

```
if model_choice == 2
   fprintf('Estimated transition time (95%% of burst phase complete): %.2f minutes\n', t_transition);
end

% Export data to CSV
experimental_data = [time', Mt'];
writematrix(experimental_data, 'experimental_data.csv');
nonlinear_fit_data = [time', Mt_predicted'];
writematrix(nonlinear_fit_data, 'nonlinear_fit.csv');
if model_choice == 2
   phase_contributions_data = [time', M_burst', M_diffusion', rate_burst', rate_diffusion'];
   writematrix(phase_contributions_data, 'phase_contributions.csv');
end

% Notify user
fprintf('\nData exported to CSV files:\n');
fprintf('- experimental_data.csv (Time, Mt)\n');
fprintf('- nonlinear_fit.csv (Time, Mt_predicted)\n');
if model_choice == 2
   fprintf('- phase_contributions.csv (Time, M_burst, M_diffusion, Rate_burst, Rate_diffusion)\n');
end

%If you use this script, please remember to cite the author and journal
%It should be noted that although the script does a good job of finding the best fit for phase proportion, manual
adjustments may be required in order to return best fit.
%article it originally came from --- J. Phipps et al., Nanoscale, 2025
```

Stability Test

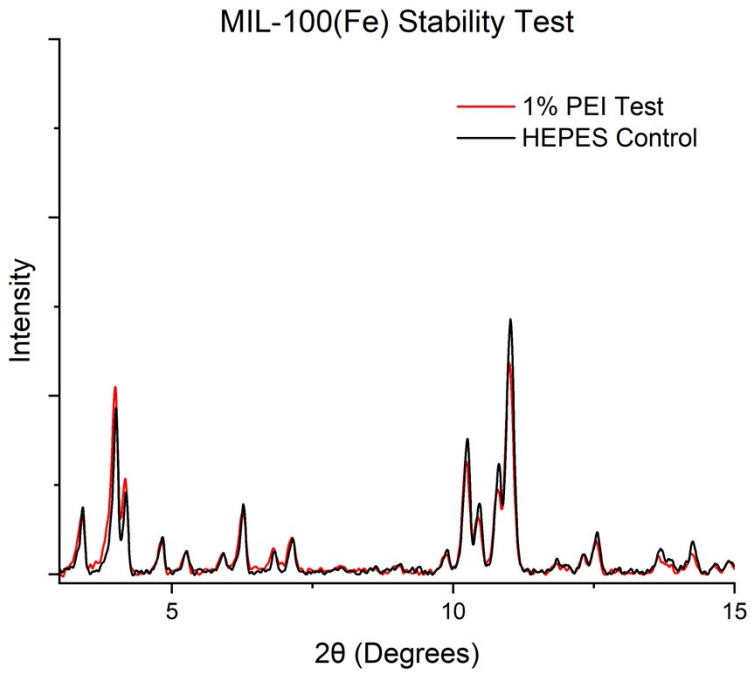


Fig. S29: pXRD after MIL-100(Fe) exposure to 1% PEI and HEPES buffer as a control.

Drug Release

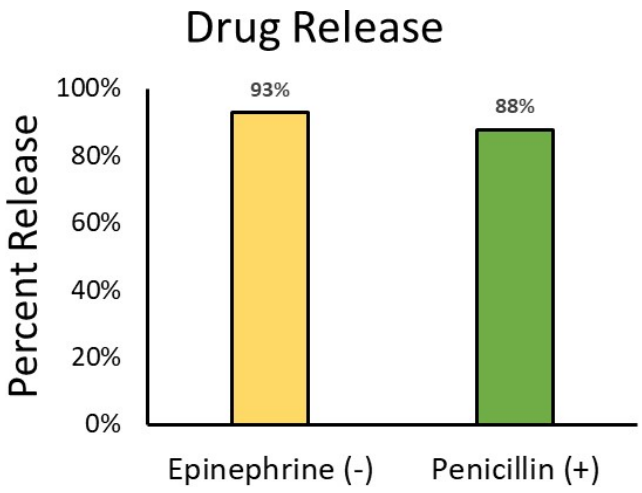


Fig. S30: Drug release from MIL-100 in the presence of 1% PEI.

Cell Morphology

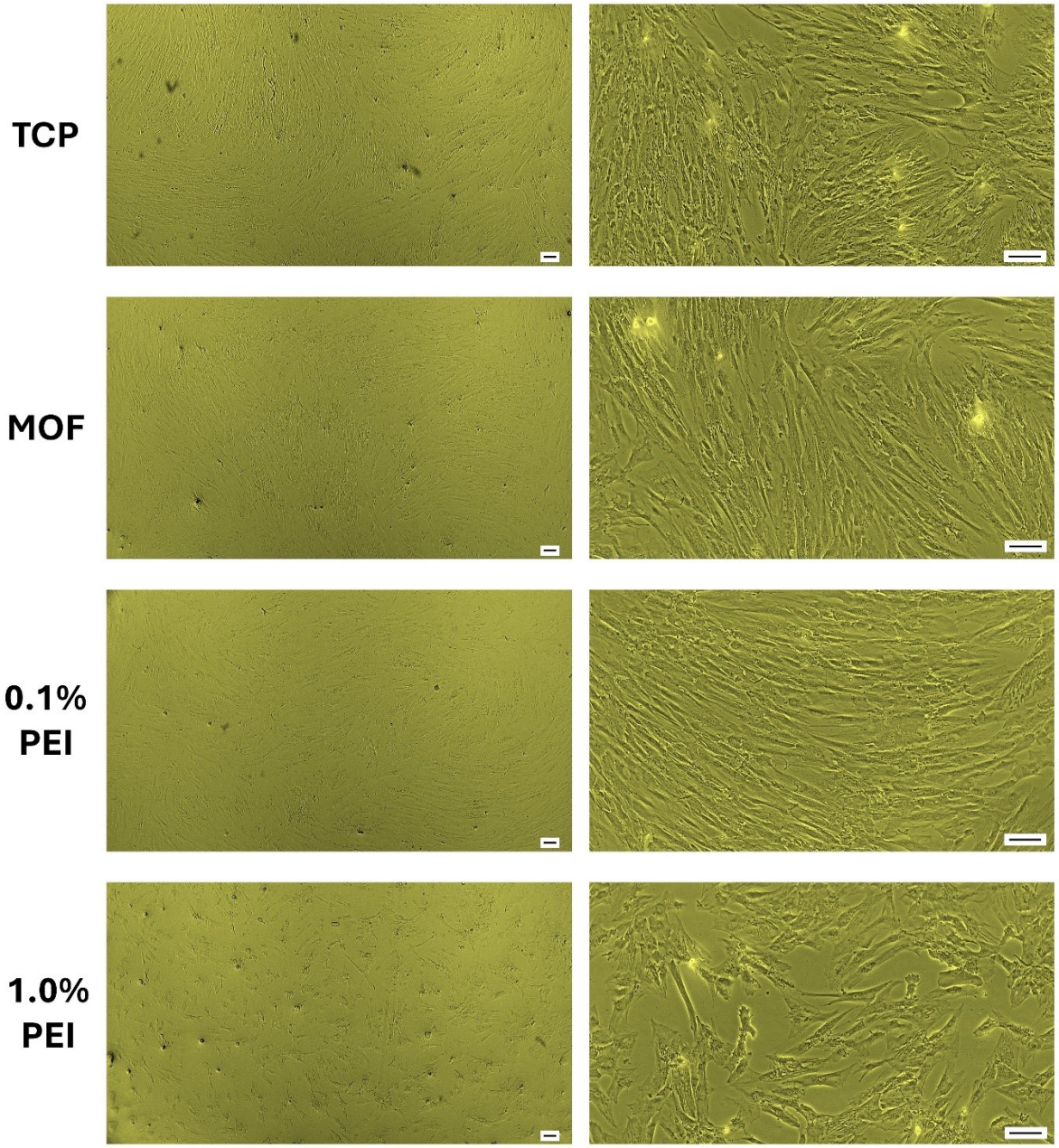


Fig. S31: bright-field microscopy images were captured to evaluate cell morphology and adhesion patterns after three days of culture (scale bars represent 100 µm).