Electronic Supplementary Material (ESI) for Soft Matter.

Supplementary Materials

Binding of Chloroaurate to Polytyrosine-PEG Micelles Leads to anti-Turkevich Pattern of Reduction

Nikolai P. Iakimov, Andrey V. Romanyuk, Irina D. Grozdova, Elisabeth A. Dets, Nikolay V. Alov, Pavel Yu. Sharanov, Sergey V. Maksimov, Serguei V. Savilov, Sergey S. Abramchuk, Alexander Ksenofontov, Elena A. Eremina, Nikolay S. Melik-Nubarov

Chemistry Department, M.V. Lomonosov Moscow State University, Leninskiye Gory 1, build. 3, GSP-1, Moscow 119991Russia

1. Synthesis and characterization of the copolymers.

Purification of Tyrosine N-carboxyanhydride was performed under inert atmosphere using the column equipped with a stopcock and sealed with septum rubber with an argon-filled balloon.

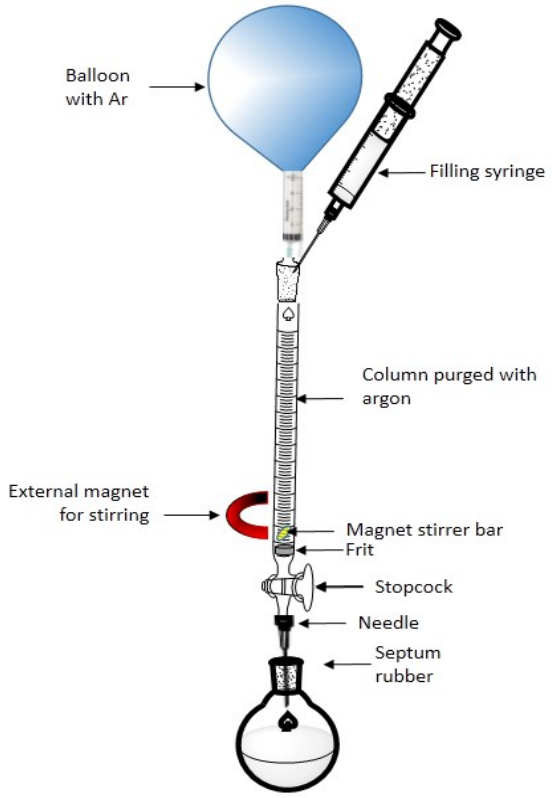


Fig. S1.Scheme of the plant used for recrystallization of Tyrosine-NCA under inert atmosphere.

Composition of the copolymers was determined with 1H-NMR spectroscopy and their molecular-weight distributions were evaluated using GPC.

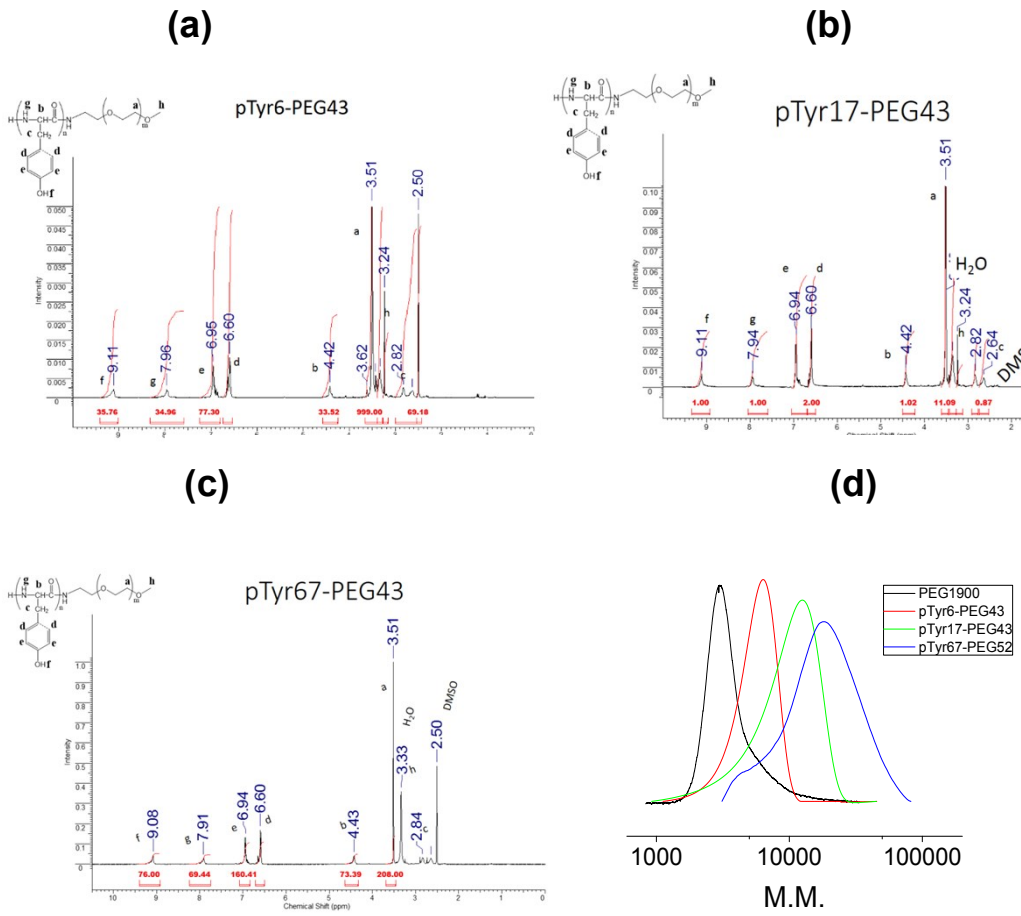


Fig. S2. Characteristics of the copolymers. 1H-NMR spectra of the copolymers pTyr6-PEG43 (a), pTyr17-PEG43 (b) and pTyr67-PEG43 (c). The copolymers were dissolved in DMSO-D6.The assignment of signals of different protons is shown in the panels. (d) GPC traces of the copolymers. The corresponding and molecular weight characteristics of the copolymers are shown in the table S1 below.

As shown in Table S1, molecular masses increased in parallel with an increase in monomer-to initiator ratio. The copolymers with low P_n (6 and 17) were narrowly distributed. However, an increase in [Tyr-NCA]/[PEG-NH_2] molar ratio resulted in broadening of molecular-mass distribution obviously due to chain transfer to Tyr-NCA as reported previously.[1] However, molecular masses determined via NMR and GPC were close to theoretical masses estimated as $P_{n,theor} = \frac{Q \cdot [TyrNCA]_0}{[PEGNH_2]_0}$, where Q - conversion of the monomer.

[1] Goodman M., Hutchinson J. The Mechanisms of Polymerization of N-Unsubstituted N-Carboxyanhydrides. J. Am. Chem. Soc. 1966, V. 88. N. 15. P. 3627-3630.

Table S1. Feed ratios and polymerization degrees of the copolymers

Copolymer	$\frac{[Tyr-NCA]_0}{[PEG-NH_2]_0}$	Conversion, %	P_{ntheor}	Mn,(GPC)	Mw (GPC)	Đ	P_n (NMR)	P_n (GPC)
pTyr6	15	40	6	6610	8540	1.29	6	7
pTyr17	29	66	19	9063	10191	1.17	17	17
pTyr67	95	85	81	13582	19942	1.47	67	44

As far as the copolymers were analyzed with GPC in DMF/0.1% LiBr at 50°C using PMMA standards for calibration, PEG and polytyrosine molecular weights appear to be affected. Corrected molecular weights can be calculated using eq. S1[2]

$$LogM_{corr} = \frac{1+\alpha'}{1+\alpha} \cdot LogM_{app} - \frac{1}{1+\alpha} Log \frac{K}{K'}, \quad \text{(S1)}$$

where K and α - are Mark-Kuhn-Houwink-Sakurada (MKHS) parameters of the analyzed polymer and K' and α' - are the corresponding parameters of PMMA standards. K and α parameters for PEG are known to be 0.024 ml/g and 0.75[3] respectively and K' and α' for PMMA are known to be 0.02094 ml/g and 0.642[4]. The corrected molecular weight of PEG-NH_2 was 1890 resulting in its polymerization degree of 43. We did not make similar corrections for the copolymers due to unavailability of MKHS parameters of polytyrosine.

[2] Y. Guillaneuf, P. Castignolles, Using Apparent Molecular Weight from SEC in Controlled/Living Polymerization and Kinetics of Polymerization. J. Polymer Sci, 2008, 46, 897-911.

[3] Polymer Handbook, 4th Ed., Edited by J. Brandrup, E. H. Immergut, and E.A.Grulke, John Wiley aαnd Sons, 1999, p. VII-32.

[4] American Polymer Standards Corporation, Providing Molecular Weight Determination Services and Polymer Standards since 1983, http://www.ampolymer.com/Mark-Houwink.html

2. Evaluation of the stoichiometry of tyrosine oxidation by chloroauric acid.

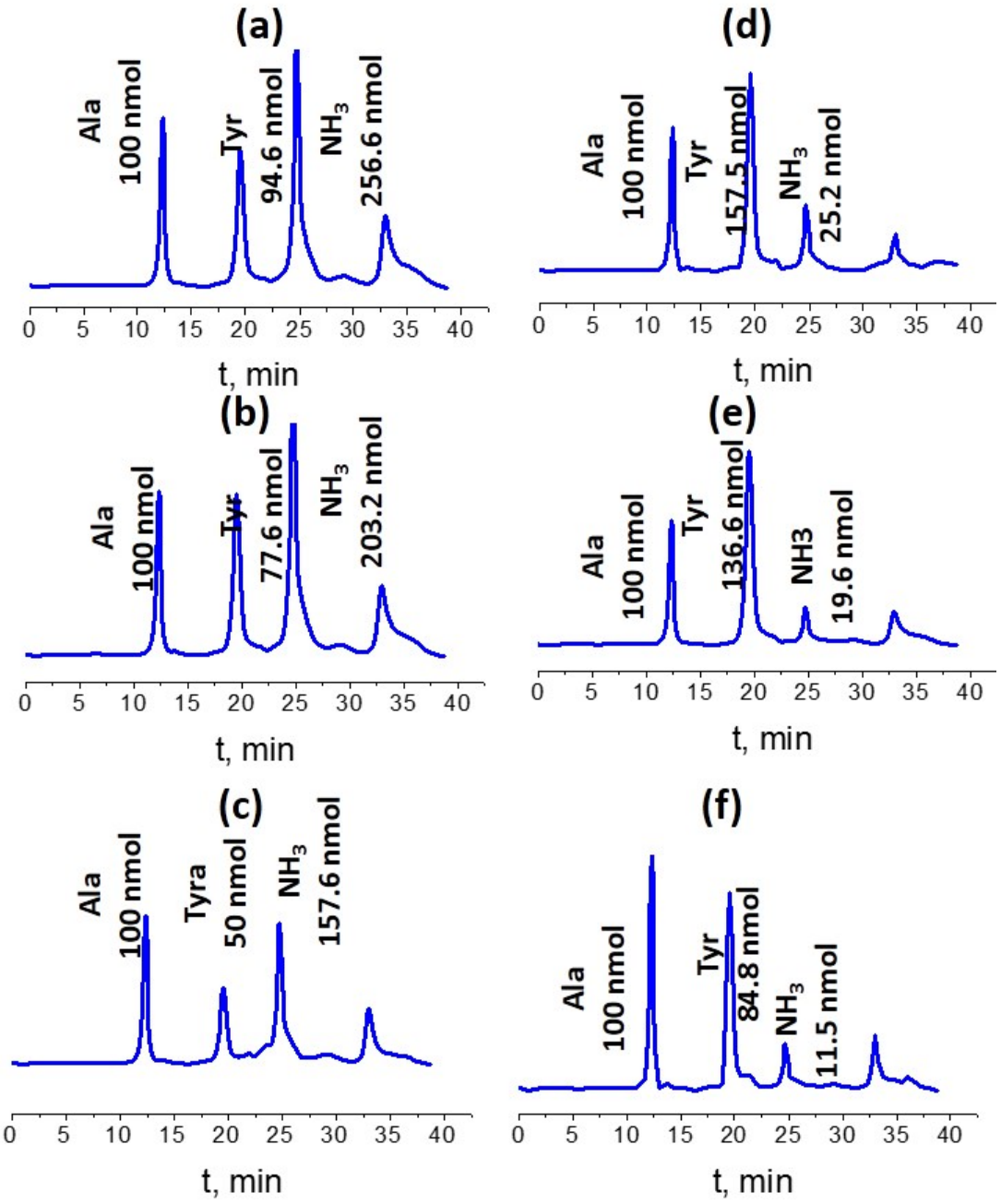


Fig. S3. Chromatographic tracesfromamino acid analysis of 0.055 mg (a), 0.046 mg (b) and 0.029 mg (c) of the composite obtained by oxidation of a of pTyr6-PEG43 block copolymer with 0.67 mM chloroauric acid and the same amounts of the untreated copolymer (d-f.). At the beginning of each experiment, 100 nmol alanine was added to the samples as the internal standard.Unidentified peak at the end of the chromatogram corresponds to the regeneration of the column.

Table S2. Evaluation of the stoichiometry of the pTyr oxidation by $HAuCl_4$

Amount of pTyr6-PEG43@Au, mg	Amount of Tyr according to amino acid analysis, nmol	Amount of consumed $HAuCl_4$, nmol	Fraction of consumed Tyr	Stoichiometry, C(oxidized Tyr)/C(Au)
0.055	94.6	21.4	0.39	2.94
0.046	77.6	18.0	0.43	3.28
0.029	50.0	11.4	0.41	3.05
Average stoichiometry				3.09±0.17

3. Spectra evolution during GNPs synthesis in the micelles of different copolymers. Effect of copolymer composition and Tyr/chloroaurate ratio.

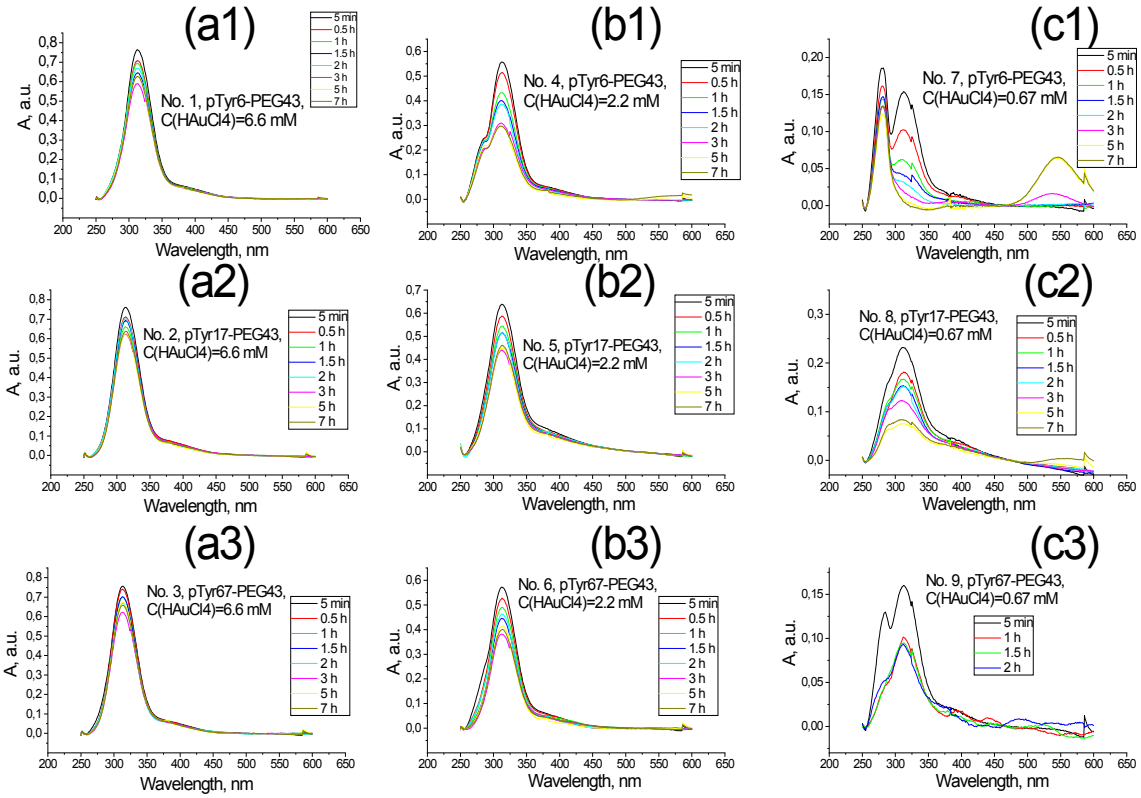


Fig. S4. UV-spectra of the reaction mixtures containing 4.3 mg/mL of pTyr6-PEG43 (a1,b1, c1), 2.83 mg/mL of pTyr17-PEG43 (a2, b2, c2), 1.93 mg/mL pTyr67-PEG43 (a3, b3, c3) and 6.67 mM $HAuCl_4$ (a1, a2, a3), 2.2 mM$HAuCl_4$ (b1, b2, b3), 0.67 mM $HAuCl_4$ (c1, c2, c3) recorded after 0.08, 0.5, 1, 1.5, 2, 3, 5 and 7 hours at 25°C.50 μl aliquots taken from the reaction mixtures were diluted 20-fold with 0.1 M HCl after above indicated time intervals.

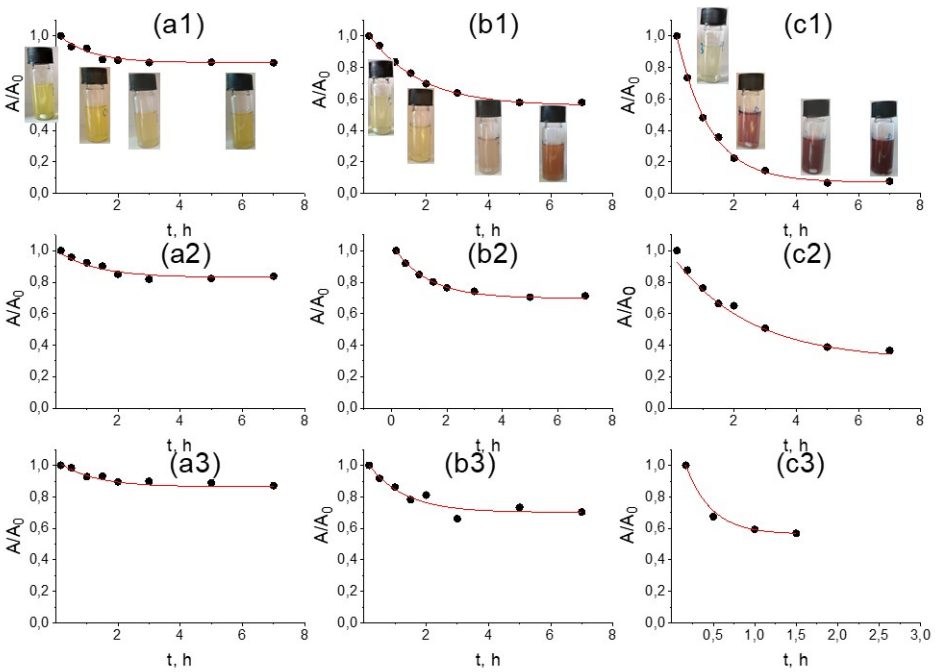


Fig. S5. Kinetics of tetrachloroaurate consumption and evolution of the appearance of reaction mixtures containing 4.3 mg/ml of pTyr6-PEG43 (a1, b1, c1), 2.83 mg/ml of pTyr17-PEG43 (a2, b2, c2), 1.93 mg/ml pTyr67-PEG43 (a3, b3, c3) and 6.6 mM $HAuCl_4$(a1, a2, a3), 2.2 mM$HAuCl_4$(b1, b2, b3), 0.67 mM $HAuCl_4$(c1, c2, c3) recorded after 0.08, 0.5, 1, 1.5, 2, 3, 5 and 7 hours at 25^{o}C.

4. Evidence for the formation of non-covalent complexes between $HAuCl_4$ and pTyr-PEG.

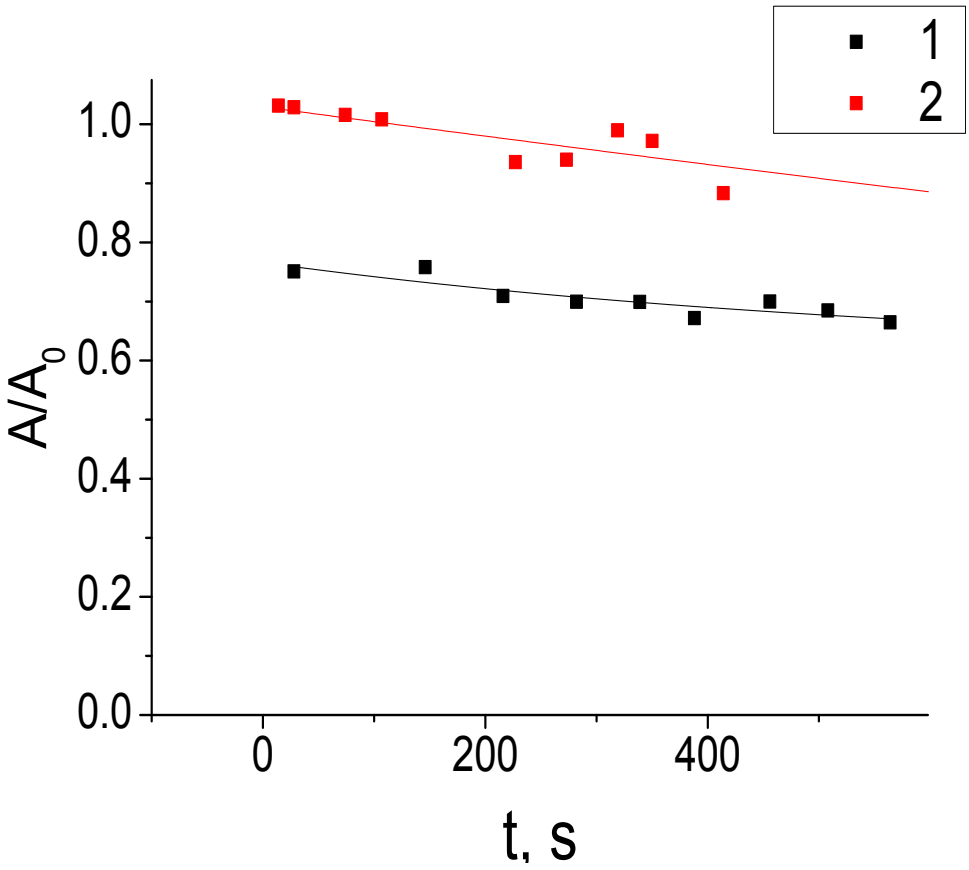


Fig. S6. Time dependence of the concentration of tetrachloroaurate in the supernatants of the reaction mixture of pTyr6-PEG43 (C(Tyr)=3.3 mM) with 0.67 mM of $HAuCl_4$ diluted 20-fold with 0.1M HCl after removal of the pellet (1). Curve (2) corresponds to the samples obtained similarly, but diluted with 0.1M HCl prior to centrifugation. The samples were taken from the reaction mixture and centrifuged immediately at 6500×g during 2 min. The x-axis corresponds to the incubation time of tetrachloroaurate with polytyrosine.

5. Size distributions of GNPs in different composites. TEM microimages of GNPs contained in different composites.

Table S3. Effect of the length of polytyrosine block and Tyr/$HAuCl_4$ ratio on GNP size. The concentration of Tyr repeat units 3.3 mM, temperature 25°C.

No.	DP of pTyr block	C($HAuCl_4$),mM	C(Au(III))/ C(Tyr)	GNP size (TEM), nm	PDI
1	6	6.7	2.0	8.0	1.16
2	17	6.7	2.0	8.8	1.14
3	67	6.7	2.0	7.5	1.14
4	6	2.2	0.67	8.1	1.09
5	17	2.2	0.67	8.0	1.08
6	67	2.2	0.67	7.0	1.11
7	6	0.67	0.2	18.8	1.63
8	17	0.67	0.2	22.8	1.35
9	67	0.67	0.2	21.3	1.74

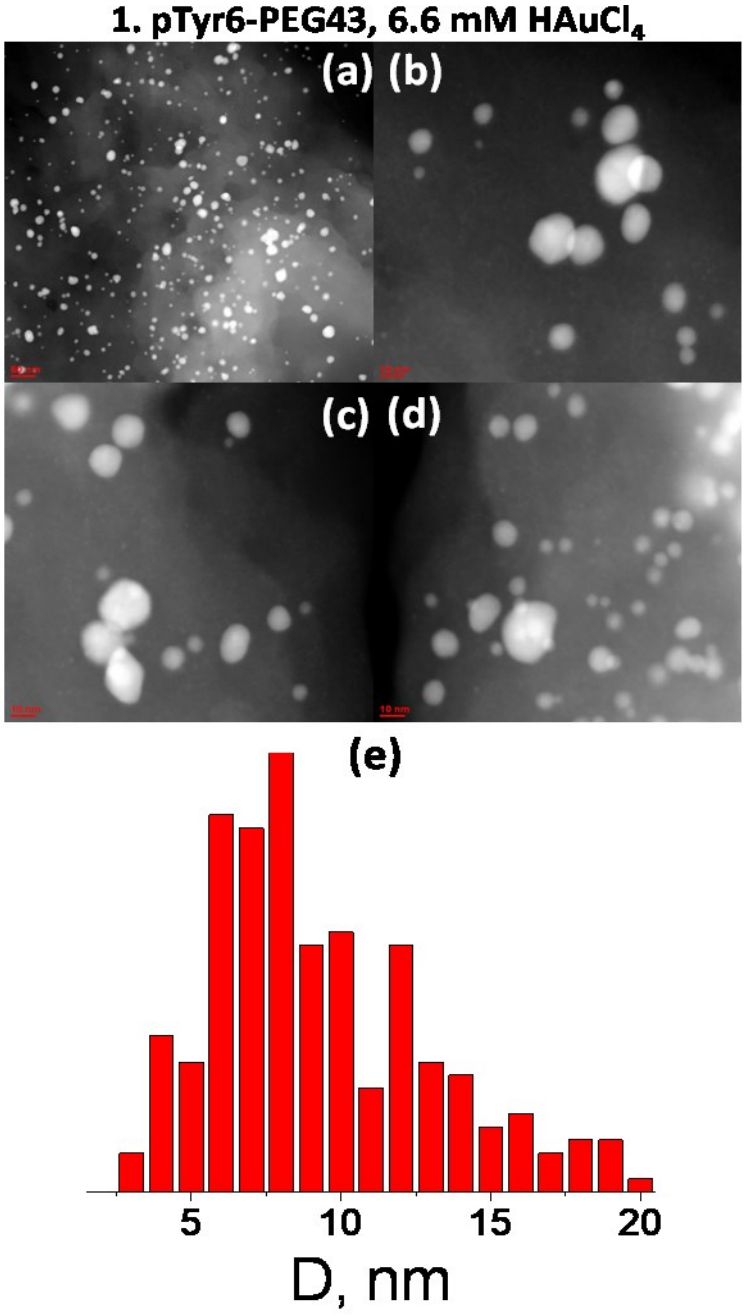


Fig. S7. Dark field HR-TEM images (a-d) and size distribution (e) of GNPs obtained during the reduction of 6.6 mM$HAuCl_4$with pTyr6-PEG43 block-copolymer atC(Tyr)=3.3 mM.

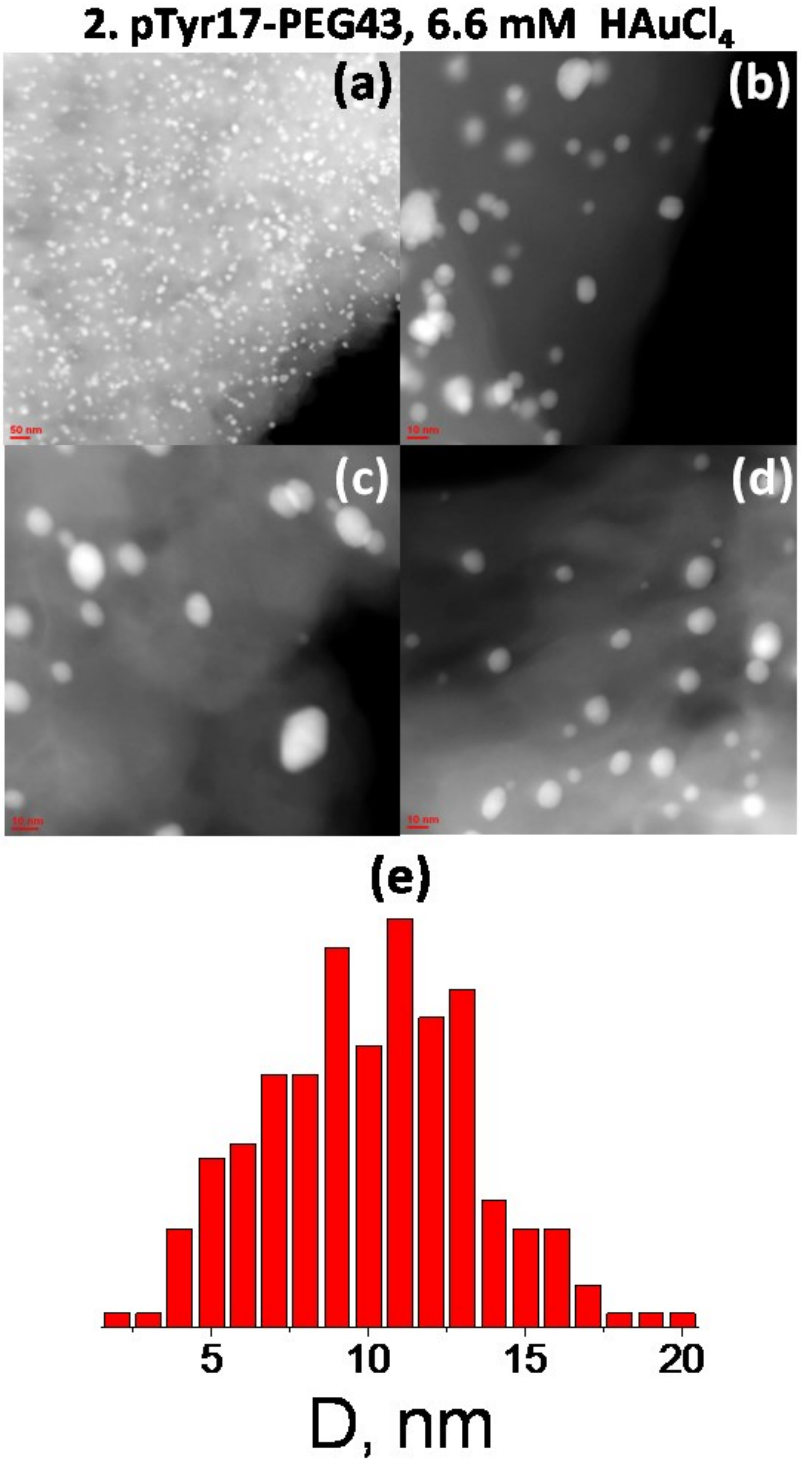


Fig. S8. Dark field HR-TEM images (a-d) and size distribution (e) of GNPs obtained during the reduction of 6.6 mM $HAuCl_4$ with pTyr17-PEG43 block-copolymer at C(Tyr)=3.3 mM.

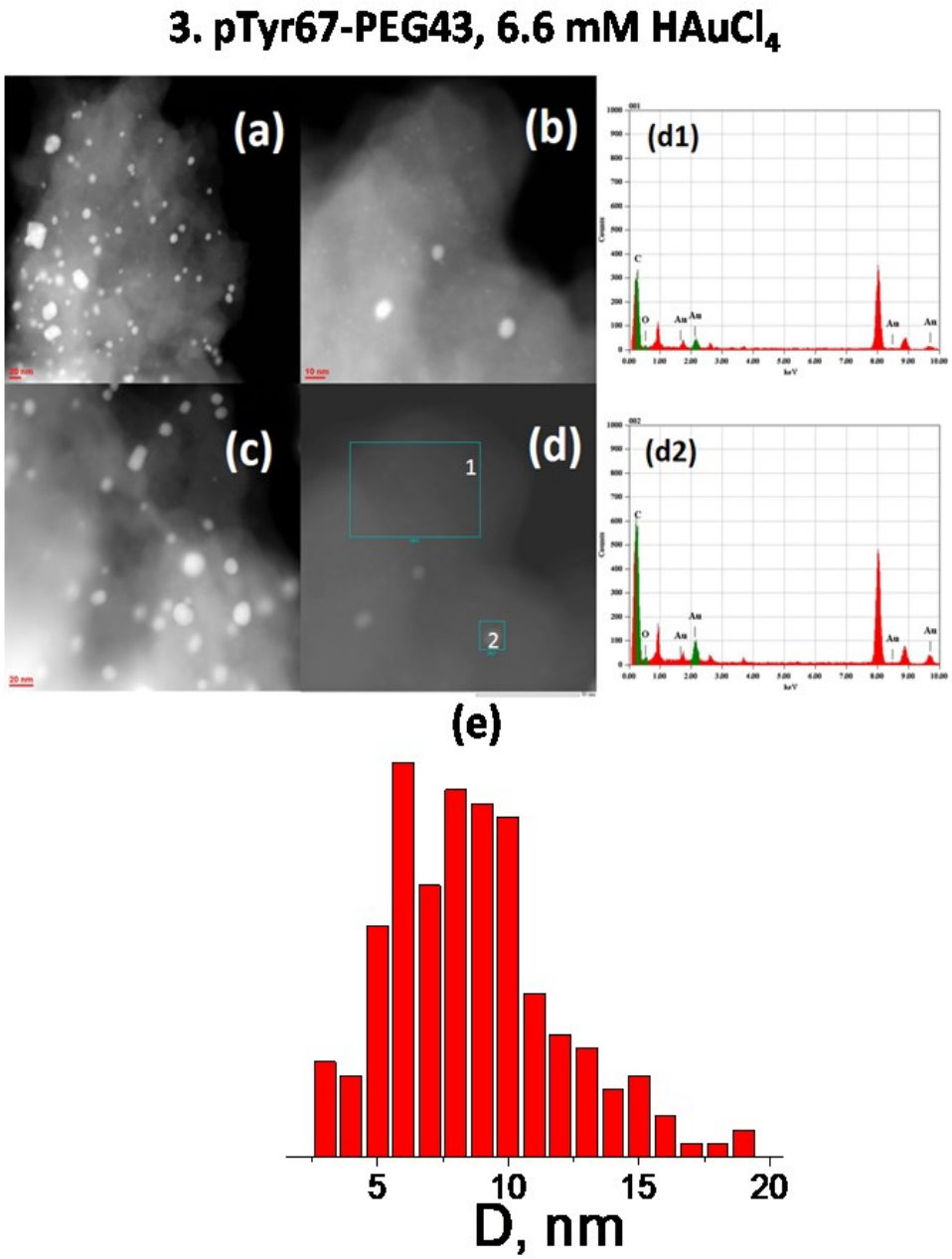


Fig. S9. Dark field HR-TEM images (a-c), and size distribution (e) of GNPs obtained during the reduction of 6.6 mM $HAuCl_4$ with pTyr67-PEG43 block-copolymer at C(Tyr)=3.3 mM. Panels (d1) and (d2) demonstrate Energy Dispersive X-ray spectra of the areas 1 and 2 in panel (d). The peaks with energies 2.12 and 9.71 keV correspond to gold(M, K_α lines). Another peaks correspond to copper(L_α(0.93 keV), K_α(8.04 keV), K_β(8.9 keV)), oxygen(K_α(0.53 keV)), carbon(K_α(0.28 keV)).

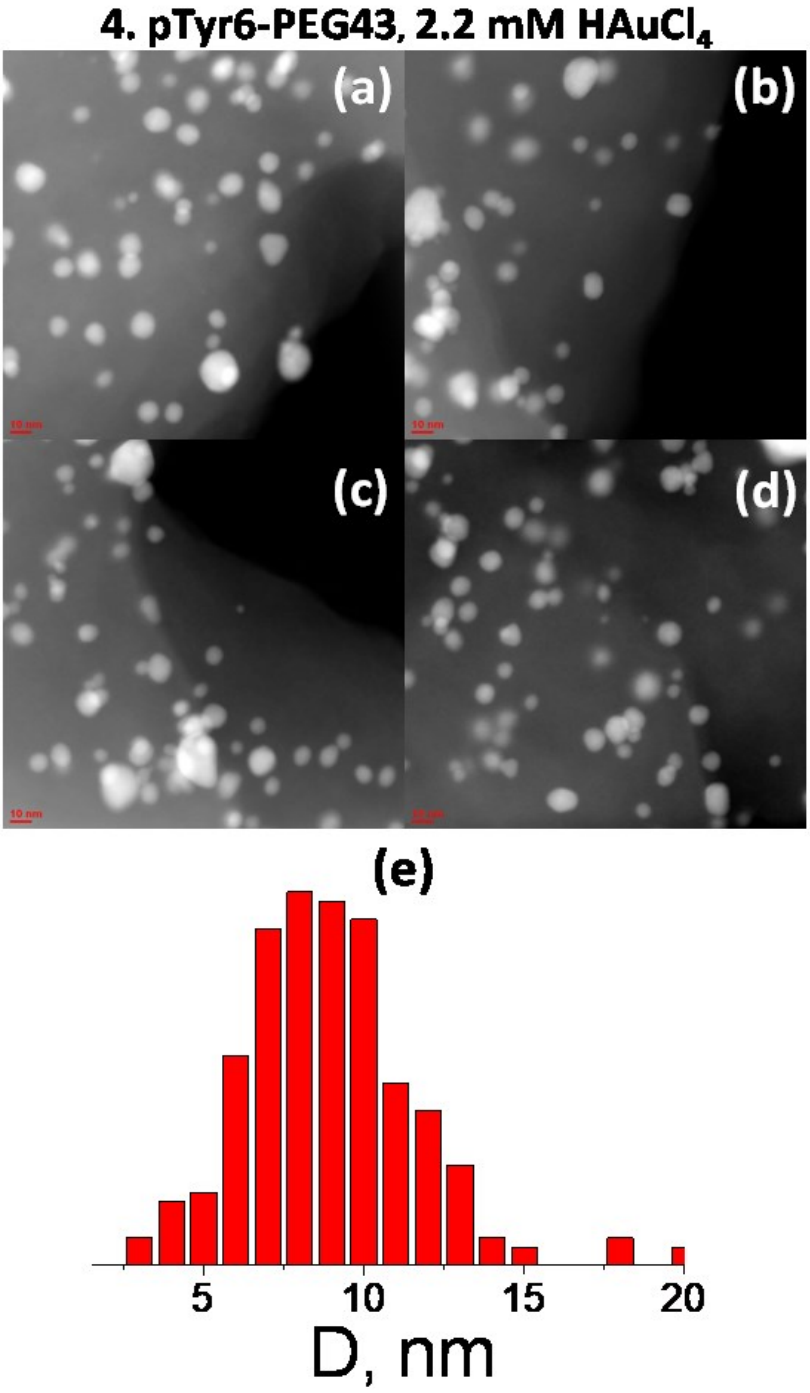


Fig. S10. Dark field HR-TEM images (a-d) and size distribution (e) of GNPs obtained during the reduction of 2.2 mM $HAuCl_4$ with pTyr6-PEG43 block-copolymer at C(Tyr)=3.3 mM.

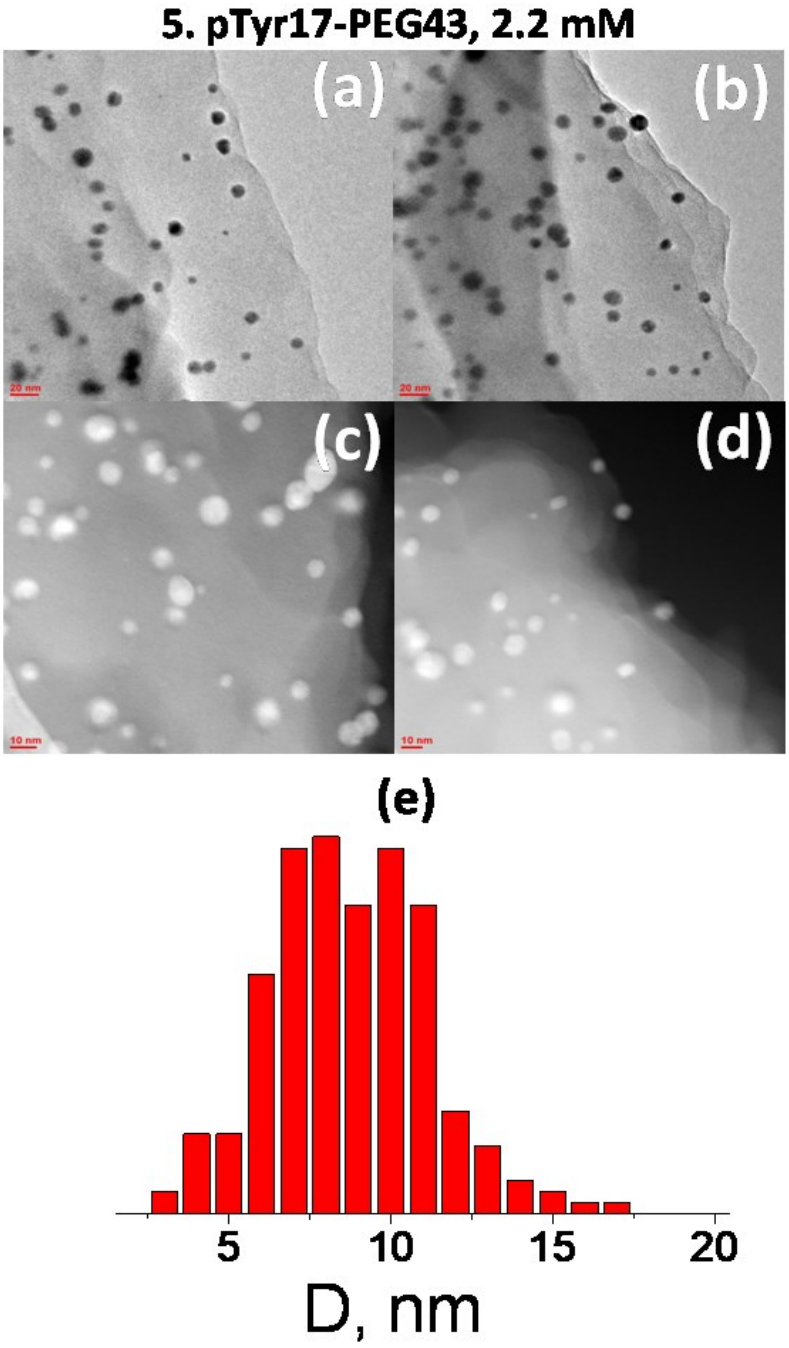


Fig. S11. Bright field (a, b) and dark field (c, d) HR-TEM images and size distribution (e) of GNPs obtained during the reduction of 2.2 mM $HAuCl_4$ with pTyr17-PEG43 block-copolymer at C(Tyr)=3.3 mM.

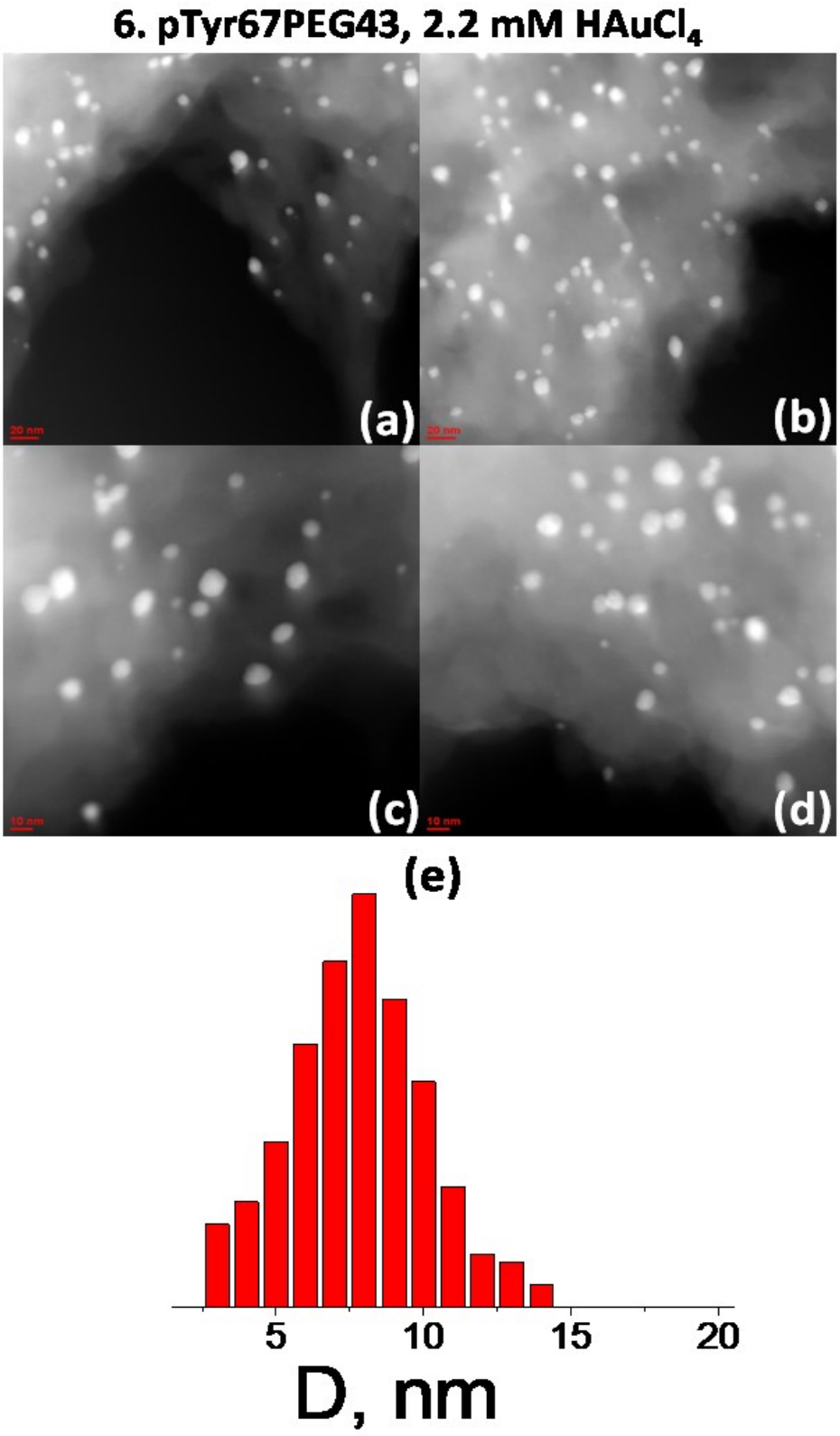


Fig. S12. Dark field HR-TEM images (a-d) and size distribution (e) of GNPs obtained during the reduction of 2.2 mM $HAuCl_4$ with pTyr67-PEG43 block-copolymer at C(Tyr)=3.3 mM.

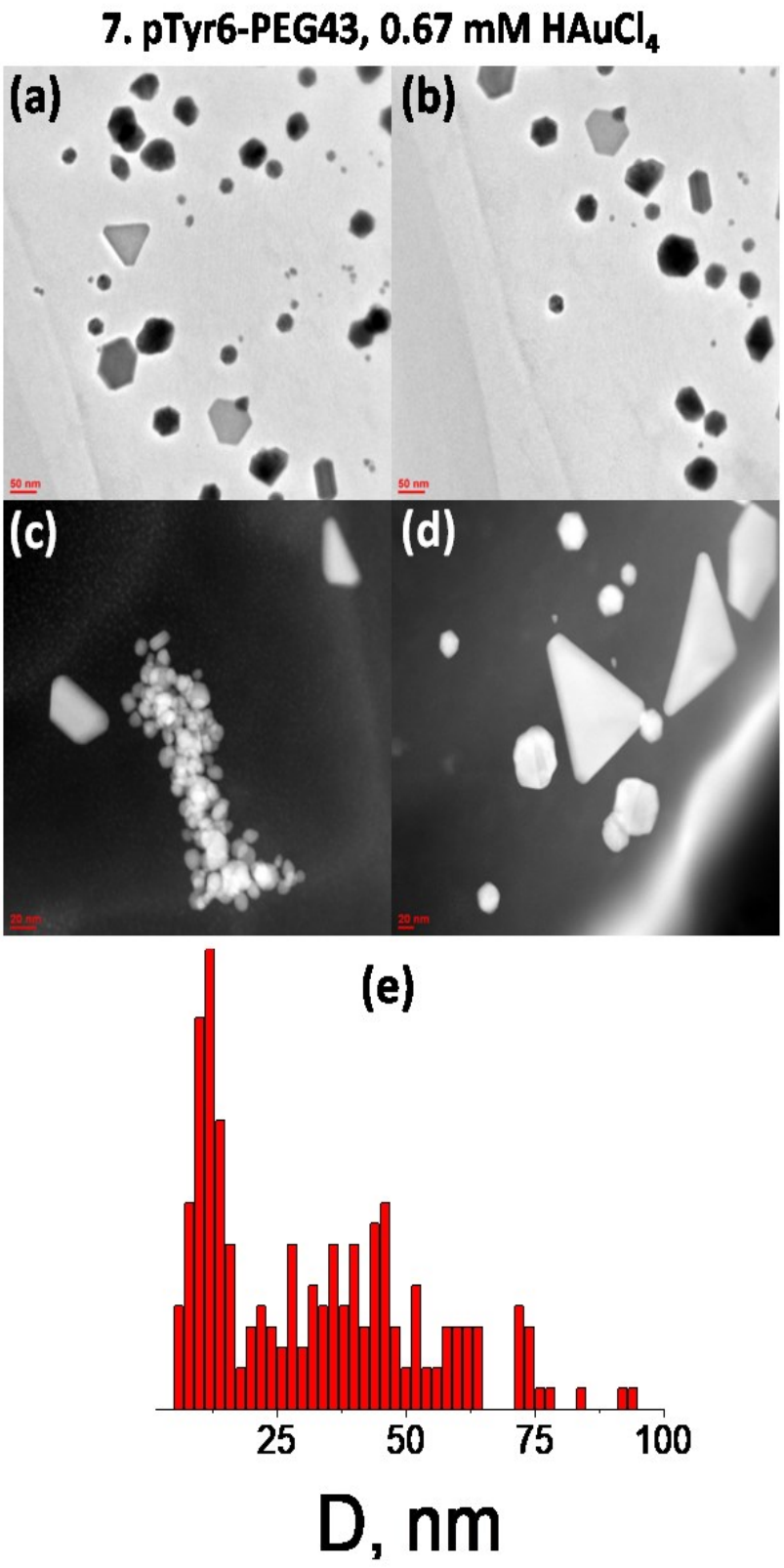


Fig. S13. Bright field (a, b) and dark field (c, d) HR-TEM images and size distribution (e) of GNPs obtained during the reduction of 0.67 mM $HAuCl_4$ with pTyr6-PEG43 block-copolymer at C(Tyr)=3.3 mM.

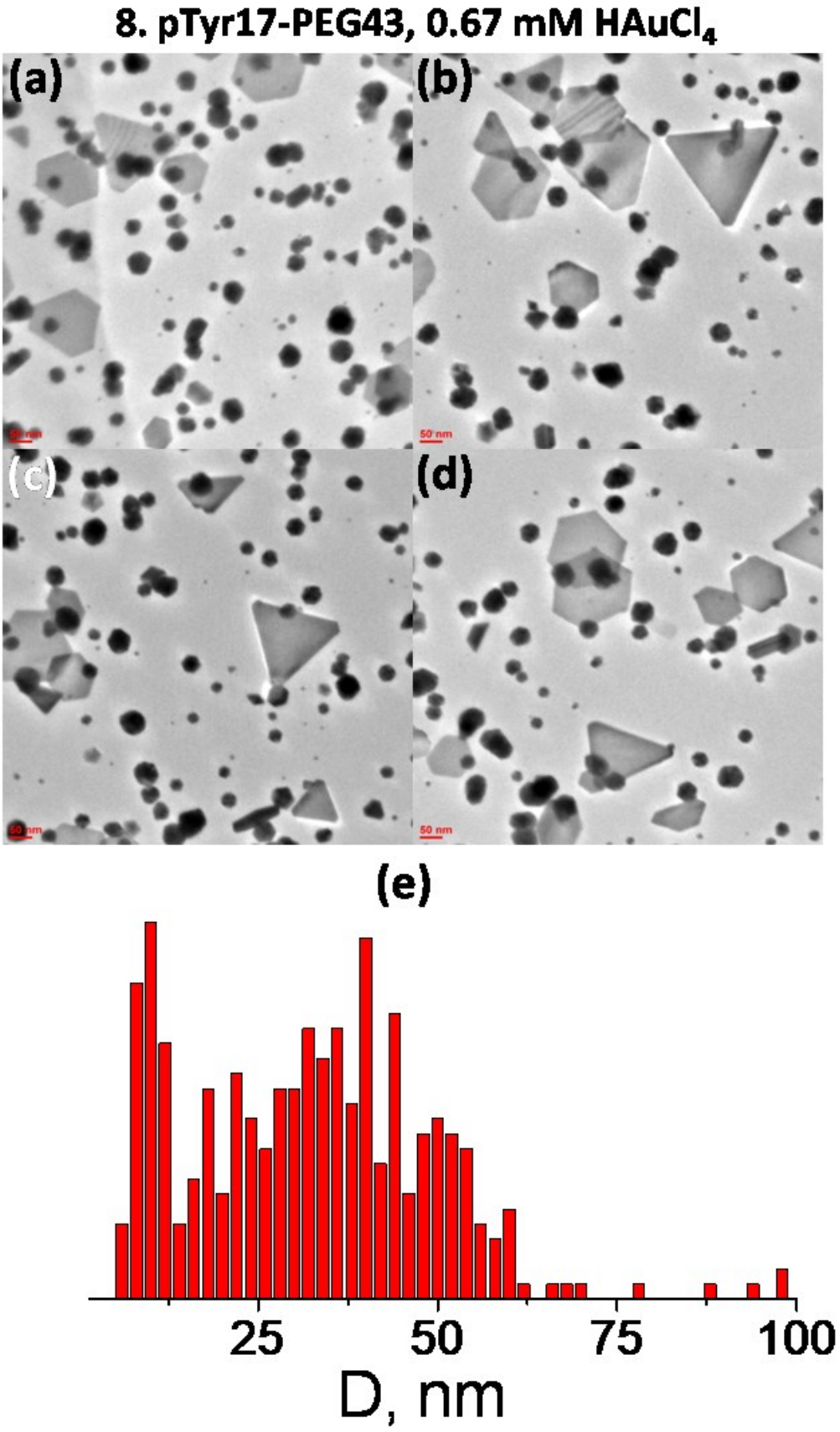


Fig. S14. Bright field HR-TEM images (a-d) and size distribution (e) of GNPs obtained during the reduction of 0.67 mM $HAuCl_4$ with pTyr17-PEG43 block-copolymer at C(Tyr)=3.3 mM.

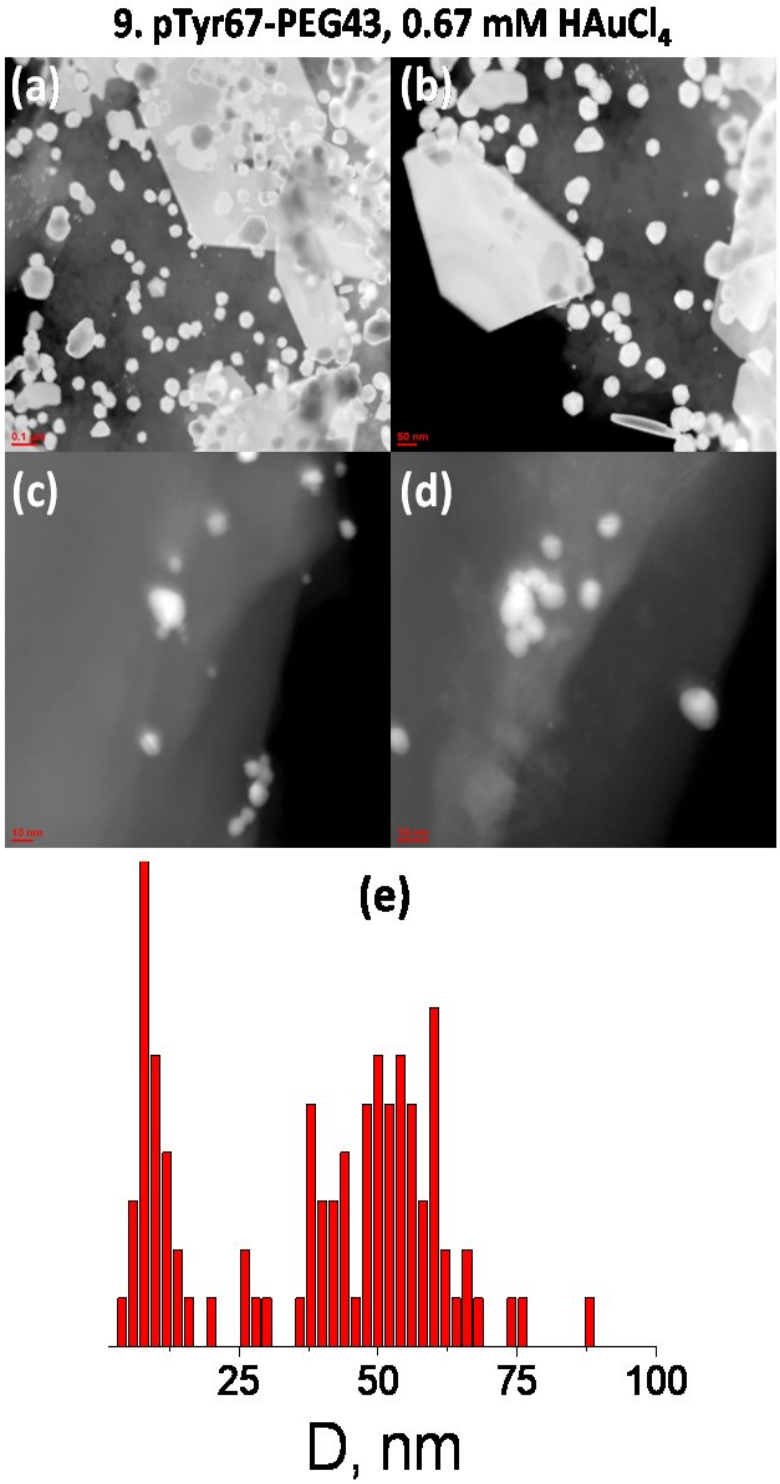


Fig. S15. Dark field HR-TEM images (a-d) and size distribution (e) of GNPs obtained during the reduction of 0.67 mM $HAuCl_4$ with pTyr67-PEG43 block-copolymer at C(Tyr)=3.3 mM.

6. Sedimentation analysis of the composites.

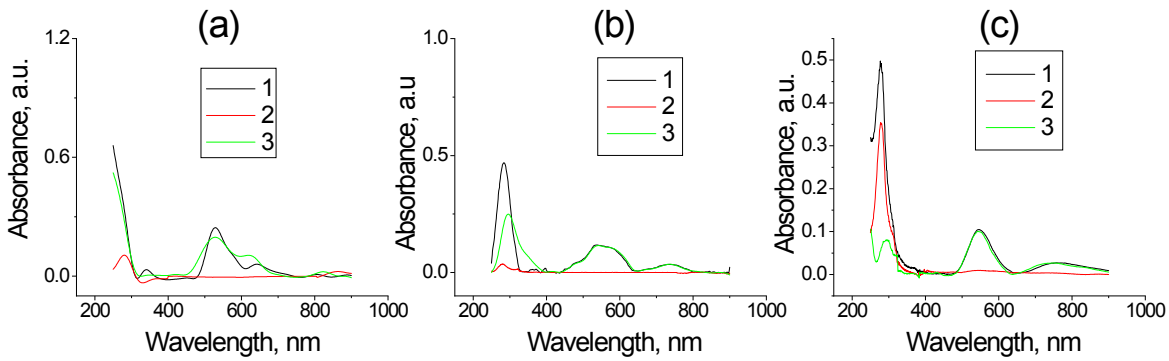
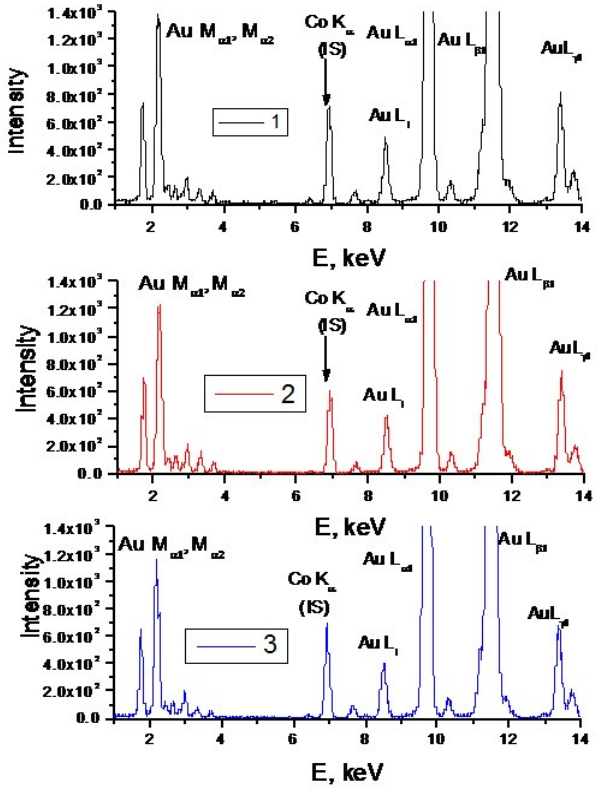
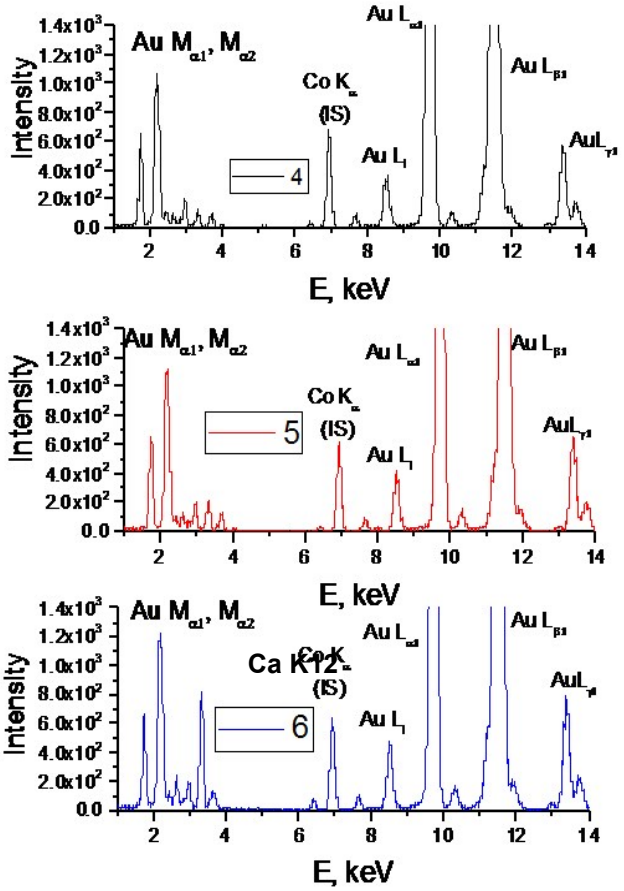


Fig. S16. UV-Vis spectra of the 0.6 mg/ml water solutions of the composites (1), the supernatants after 10 min centrifugation at 12000g (2) and the pellets resuspended in the initial volume of 0.1 M NaOH (3). The composites refer to Nos. 1 (a), 4 (b) and 7 (c) described in Table 2 of the Main Text. They were prepared from 3.3 mM pTyr6-PEG43 copolymer at $HAuCl_4$ concentrations of 6.7 mM (a), 2.2 mM (b), and 0.67 mM (c) correspondingly.

7. TRXF spectra of the composites.




Intensity
E, keV
Au M$_{\alpha1}$, M$_{\alpha2}$
Co K$_{\alpha}$ (IS)
Au L$_{\alpha1}$
Au L$_{\beta1}$
Au L$_{l}$
AuL$_{\gamma1}$
4
5
6
Ca K12

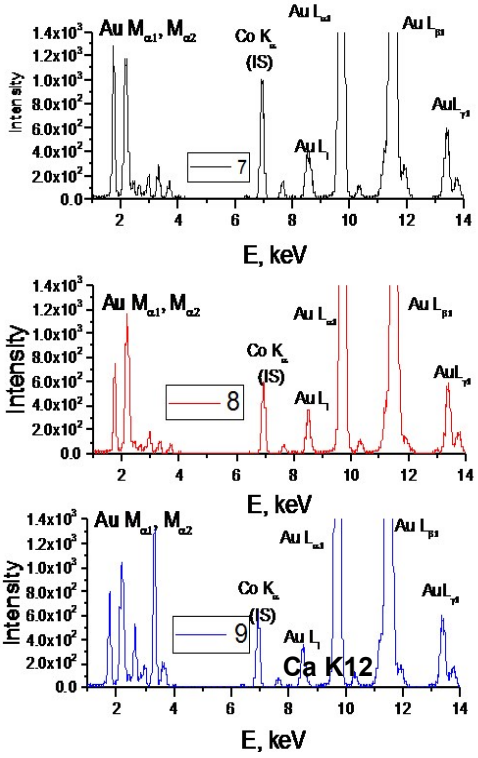

Intensity
E, keV
Au M$_{\alpha1}$, M$_{\alpha2}$
Co K$_{\alpha}$ (IS)
Au L$_{\alpha1}$
Au L$_{\beta1}$
Au L$_{l}$
AuL$_{\gamma1}$
7
8
9
Ca K12

Fig. S17. Total Reflection X-ray Spectra of the composites No. 1-9 described in Table 3. The spectra corresponding to the composites based on pTyr6-PEG43 are presented as black, pTyr17-PEG43 – red and pTyr67-PEG43 – blue. Assignment of the bands is presented in the panels. Cobalt was used as an internal standard.

7. Reduction of 4-nitrophenol by sodium borohydride catalyzed by pTyr-PEG@Au composites.

Table S4. Concentration of gold in 1.25 mg/mL solutions of the composites determined with TXRF, surface area of GNPs in the composites at a constant concentration corresponding to 4.18 μM of Au and catalytic properties of the composites measured at this concentration of GNPs.

Sample No.	C(Au), μM (TXRF)	Sample		S, m^2/L	$k_{cat}\times S$, $m^2\cdot mol^{-1}\cdot s^{-1}$
		DP(pTyr)	$\frac{[Tyr]}{[HAuCl_4]}$		
1	635±5	6	0.5	0.032±0.003	4.87±0.72
2	482±4	17	0.5	0.036±0.003	7.19±0.21
3	309±2	67	0.5	0.034±0.003	5.39±0.42
4	674±6	6	1.5	0.025±0.003	3.26±0.24
5	602±6	17	1.5	0.032±0.003	2.35±0.48
6	556±5	67	1.5	0.037±0.004	3.22±0.33
7	581±5	6	5	0.017±0.004	2.10±0.63
8	713±6	17	5	0.023±0.004	2.16±0.47
9	557±5	67	5	0.019±0.004	1.65±0.10

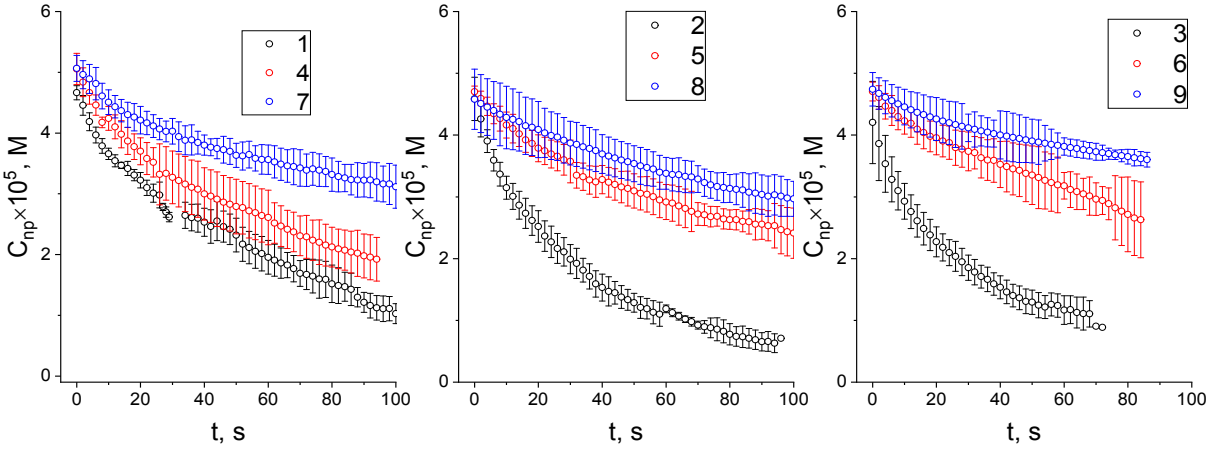


Fig. S18. Kinetics of 4-nitrophenol decay in the presence of 30 mM sodium borohydride catalyzed by pTyr-PEG@Au composites synthesized by oxidation of pTyr6-PEG43(a), pTyr17-

PEG43 (b) and pTyr67-PEG43 (c) at $[Tyr]/[AuCl_4^-]$ molar ratio of 0.5 (samples 1-3), 1.5 (samples 4-6) and 5 (samples 7-9). Solid lines correspond to numerical solution of eq. 7 produced by the script written in Python 3.2.2. (See section 8 below)

8. Python 3 script for the numerical solution of differential equation (7) which describes the kinetics of catalytic reduction of 4-nitrophenol by sodium borohydride.

```
"""
The script works in several steps:
1) Necessary libraries are imported, the default directories are set up
2) Default values for the parameters are set up
3) Looks for data files in the subdirectory /raw, the names of the files should end with _SS.txt to get recognized and
processed
4) Reads the data files line by line and extracts the data according to the pattern of data files:
   a) Block 1: parameter name and its value separated by a tab or a space
   b) Separator line: 20 dash symbols
   c) Block 2: the data starting with the line of headers
5) Groups the data sets by samples (all replicats of the same sample belong to one group)
6) The system of ODEs and the objective function (the one that gets minimized by the solver) are described
7) For each data set fitting is set up and the optimal values of the parameters are found
8) Both input and fitted data are plotted
   A) For individual samples
   B) For grouped samples
9) Both input and fitted data are exported to a txt file
   A) For individual samples
   B) For grouped samples
10) All rate constants are exported to a txt file

"""

# STEP_1
import os, re
import numpy as np
from scipy import integrate
import pylab as plt
from lmfit import Parameters, minimize, report_fit

# the directory of the script is the working directory
script_folder = os.path.dirname(os.path.abspath(__file__))
# the data files are expected in the subdirectory /raw
if os.path.isdir(script_folder + '/raw'):
   print('The directory for files with raw data was found')
else:
   print('No directory for files with raw data was found')

# the directory where to look for files
list_f = os.listdir(script_folder + '/raw')


# STEP_2
"""
```

```
Definition of the default values for the parameters
These values are used if not imported from the files containing data
"""
Knp_init, Kbh_init, k_init, n_init, m_init = 6922, 69, 1.6e-4, 0.6, 1.0
Cbh, S = 0.03, 1.0  # these default values will be replaced for the ones read from the data files

# STEP_3
# dictionary to store the input data
samples_dict = {'Filenames':[], 'Samples':[], 'Groups':[]}

"""
Files ending with "SS_raw.txt" will be recognized as data files and then processed (SS stands for steady state)
Also, the file name is expected to start with sX where X is one or several digits.
Therefore, the file name should have the following format: "sXX(any characters)_SS_raw.txt"
"""
for file in list_f:
    if file.endswith("_SS_raw.txt") or file.endswith("_SS_raw.TXT"):
        filename_w_ext = os.path.basename(file)
        samples_dict['Filenames'].append(filename_w_ext)

print('The following data files will be processed using the model for the stationary kinetics:')
for f_name in samples_dict['Filenames']:
    print(f_name)

# STEP_4
# reading all the data files and extracting the data
for file in samples_dict['Filenames']:
    sample = {}  # dictionary to store data of a single curve
    print('Processing file...\t', file)
    ts = []  # a list of time points in seconds
    c = []  # a list of experimental C(NP) values
    param_list = [Cbh, S] # a list of default values
    param_upd = [0, 0]  # flags: 0 -- updated, 1 -- not updated
    param_names = ['Cbh', 'S']  # names of the parameters that will be searched for in the file
    # the full list of parameters can be read from the file:
    # param_names = ['Knp_init', 'Kbh_init', 'n_init', 'm_init', 'k_init', 'Cbh', 'S']

    # opening files with UTF-8 enconding
    with open(script_folder + '/raw/' + file, 'r', encoding="utf8") as input_file:
        lines = input_file.readlines()  # read all lines and make them a list of lines

    # looking for the line separating the block of parameters from the block of data points
    for j in range(len(lines)):
        if re.fullmatch('-{20}\s+', lines[j]):  # the line should match the pattern precisely
            file_divider = j
            break
    # extracting parameters from the block before the separating line
    for j in range(0, file_divider, 1):
        for i in range(len(param_list)):
            if re.match(param_names[i] + '\s+', lines[j]) and param_upd[i] == 0:
                param_list[i] = float(re.split('\s+', lines[j])[1])
                param_upd[i] = 1
    # use the values from the file to update the parameters
    Cbh, S = [param_list[i] for i in range(len(param_list))]

    # Extraction of data after the divider and the headers line.
```

```
    # Each line with data should have the values for time point and concentration devided by a tab or a space.
    for l in lines[file_divider+2:]:
        if re.match('\d+(\.\d+)?\s\d+[\.\,]\d+E?-?\d+\s?', l):
            if l.find(',') != -1:  # if commas are used as a decimal point, they are replaced for dots
                l = l.replace(',', '.')
            split_line = re.split("\s", l)
            ts.append(float(split_line[0]))
            c.append(float(split_line[1]))
    # the data for each sample is stored as a separate dictionary
    sample = {'Title': file, 't': ts, 'c': c, 'X0': c[0], 'Knp': Knp_init, 'Kbh': Kbh_init, 'n': n_init, 'm': m_init,\
              'Cbh': Cbh, 'S': S}
    samples_dict['Samples'].append(sample)
    print('Parameters used (Knp, Kbh, n, m, Cbh, S, X0): ',\
          sample['Knp'], sample['Kbh'], sample['n'], sample['m'], sample['Cbh'], sample['S'], sample['X0'])

# STEP_5
# grouping the samples
for s in samples_dict['Samples']:
    # the file name is expected to have a name of the format sX where X is one or several digits
    if re.match('s\d\S+', s['Title']):
        g_name = re.split('_', s['Title'])[0]
        if g_name not in [g['Group name'] for g in samples_dict['Groups']]:
            samples_dict['Groups'].append({'Group name': g_name, 'Files': []})
            samples_dict['Groups'][-1]['Files'].append(s)
        else:
            samples_dict['Groups'][-1]['Files'].append(s)

# calculating the mean and SD values within each group
for group in samples_dict['Groups']:
    group['t'] = sorted(list(set(np.concatenate([file['t'] for file in group['Files']]))))
    for par in ['Knp', 'Kbh', 'n', 'm', 'Cbh', 'S']:
        group[par] = list(set([file[par] for file in group['Files']]))
        assert(len(group[par]) == 1), f'{par} values are not consistent within a group of replicates'
        group[par] = group[par][0]
    group['Data'] = {}  # will contain data  from replicates for each time point
    group['k_all'] = []  # list of rate constants obtained for replicates within one group
    group['Mean+SD'] = {'t': [], 'Mean': [], 'SD': []}
    for tt in group['t']:
        group['Data'][tt] = []  # Conc go into the 0th list, SDs go into the 1st one
        for file in group['Files']:
            if tt in file['t']:  # if there is a conc value at this time point
                group['Data'][tt].append(file['c'][list(file['t']).index(tt)])  # add this conc value to the list
        # if there are more than 1 conc value at this time point,
        # then we can average them out  and calculate SD
        # single values are ignored
        if len(group['Data'][tt]) > 1:
            group['Mean+SD']['t'].append(tt)
            group['Mean+SD']['Mean'].append(np.mean(group['Data'][tt]))
            group['Mean+SD']['SD'].append(np.std(group['Data'][tt]))
g_num = len(samples_dict['Groups'])  # the number of groups


# STEP_6
# the system of ODEs
def dX_dt(X, t, Knp, Kbh, k, n, m, Cbh, S):
    Cnp = X
```

```
    dxdt = -(k * S * ((Kbh * Cbh) ** m) * ((Knp * Cnp) ** n)) / (1 + ((Kbh * Cbh) ** m) + ((Knp * Cnp) ** n)) ** 2
    return dxdt

# the objective function
def ode_fit(params, t, data):
    solutions = integrate.odeint(dX_dt, y0=F['X0'], t=F['t'], args=(
    params['Knp'], params['Kbh'], params['k'], params['n'], params['m'], params['Cbh'], params['S']))[:, 0]
    resids = solutions - data  # return a 1D-array of residuals
    return resids

# STEP_8A
# plotting the results
plt.figure('Data fitting: Individual datasets')  # figure #1
plt.clf()
plt.suptitle('$K_{NP} = ' + f'{Knp_init:.2E}' + '\ [L/mol],\ K_{BH} = ' + f'{Kbh_init:.2E}' + '\ [L/mol]$', usetex = True,
fontsize='18') # title for the plot
plt.ylabel('Cnp')  # title for the Y-axis
marker_style = dict(linestyle=' ', marker='o', markersize=6, markeredgewidth=1.5, fillstyle='none')
col_list = ['b', 'g', 'k', 'm', 'c']  # list of colors used to plot replicates

for g, group in zip(range(0, g_num), samples_dict['Groups']):
    print('Group\t' + group['Group name'] + '\n')
    plt.subplot(1, g_num + 1, g + 1)  # plot different concentrations in a separate subplot

    for F, clr in zip(group['Files'], col_list):
        # STEP_7
        # setting up the fitting
        fit_params = Parameters()
        # each parameter can take only positive values
        min_bnd = 0.0  # the lower bound for parameters
        max_bnd = np.inf  # the upper bound for parameters
        fit_params.add('Knp', value=F['Knp'], vary=False, min=min_bnd, max=max_bnd)
        fit_params.add('Kbh', value=F['Kbh'], vary=False, min=min_bnd, max=max_bnd)
        fit_params.add('k',  min=min_bnd, max=max_bnd, vary=True)
        fit_params.add('n', value=F['n'], min=min_bnd, max=max_bnd, vary=False)
        fit_params.add('m', value=F['m'], min=min_bnd, max=max_bnd, vary=False)
        fit_params.add('Cbh', value=F['Cbh'], vary=False)
        fit_params.add('S', value=F['S'], vary=False)

        # minimize the residuals
        result = minimize(ode_fit, params=fit_params, args=(F['t'], F['c']), method='least_squares', verbose=1)
        # show the report
        result.params.pretty_print()  # detailed fitting report
        report_fit(result)

        lmfit_params = [param.value for name, param in result.params.items()]  # a list of values of the parameters
        F['k'] = result.params['k'].value
        group['k_all'].append(F['k'])

        legend_params = f'$k_{(g+1)}' + f" = {F['k']:.2E}" + '\ [mol/m^2*s]$'
        F['Model'] = integrate.odeint(dX_dt, y0=F['X0'], t=F['t'], args=tuple(lmfit_params))[:, 0]

        # STEP_8A
        plt.plot(F['t'], np.array(F['c']), color=clr, **marker_style)
        plt.plot(F['t'], np.array(F['Model']), '--',  color=clr, label=legend_params,linewidth=2, markersize=5)
```

```
        # STEP_9A
        if not os.path.isdir(script_folder + '/model'):  # check if the subdirectory /model exists
            os.mkdir(script_folder + '/model')  # create the subdirectory /model for the output data
            print(r'The directory /model for output files was created')

        with open(script_folder + '/model/' + F['Title'][:-7] + 'model.txt', 'w', encoding="utf8") as output_file:
            output_line = 'Time [s]' + '\t' + 'Conc [M] (experiment)'+ '\t' + 'Conc [M] (Model)' + '\n'
            for e1, e2, e3 in zip(F['t'], F['c'], F['Model']):
                output_line += str(e1) + '\t' + str(e2) + '\t' + str(e3) + '\n'
            output_file.write(output_line)

    # STEP_8A
    group['k_mean'] = np.mean(group['k_all'])
    group['k_SD'] = np.std(group['k_all'])
    g_params = [group['Knp'], group['Kbh'], group['k_mean'], group['n'], group['m'], group['Cbh'], group['S']]
    group['Model'] = integrate.odeint(dX_dt, y0=group['Mean+SD']['Mean'][0], t=group['Mean+SD']['t'],
                                      args=tuple(g_params))[:, 0]
    label = f"$Mean\ k={group['k_mean']:.2e} \pm {group['k_SD']:.2e}$"
    plt.plot(group['Mean+SD']['t'], group['Model'], 'r--', label=label, linewidth=2, markersize=5)

    # settings for the figure #1
    plt.ylim((1e-6, 5.0e-5))
    plt.xlim((-5.0, 150.0))
    plt.xticks(np.arange(0.0, 151.0, 50))
    if g == 0:
        plt.ylabel('$C_{NP},\ [mol/L]$', fontsize=18)
    else:
        plt.tick_params(labelleft=False)
    # plt.yscale('log')  # optional log scale
    plt.tick_params(labelsize=14)
    if g == g_num//2:
        plt.xlabel('$Time\ [s]$', fontsize=18)  # title for the X-axis
    plt.legend(loc='upper right', prop={'size': 12})
# plt.show()  # show the graph

# STEP_8B
plt.figure('Data fitting: Grouped datasets')  # figure #2
plt.clf()
plt.suptitle('$K_{NP} = ' + f'{Knp_init:.2E}' + '\ [L/mol],\ K_{BH} = ' + f'{Kbh_init:.2E}' + '\ [L/mol]$', usetex=True,
             fontsize='18')  # title for the plot
plt.ylabel('Cnp')  # title for the Y-axis
for g, group in zip(range(0, g_num), samples_dict['Groups']):
    plt.subplot(1, g_num + 1, g + 1)  # plot different concentrations in a separate subplot
    group_legend = ''
    g_params = [group['Knp'], group['Kbh'], group['k_mean'], group['n'], group['m'], group['Cbh'], group['S']]
    for i, par in zip(g_params, ['Knp', 'Kbh', 'k', 'n', 'm', 'Cbh', 'S']):
        group_legend += par + ':  ' + f'{i:.2E}\n'
    plt.errorbar(group['Mean+SD']['t'], group['Mean+SD']['Mean'], yerr=group['Mean+SD']['SD'],
                 label=group['Group name'] + ' Mean', c='c', marker='o')
    plt.plot(group['Mean+SD']['t'], group['Model'], 'r--', label='Fit\n' + group_legend, linewidth=2, markersize=5)
    # settings for the figure 2
    plt.ylim((1e-6, 5.0e-5))
    plt.xlim((-5.0, 150.0))
    plt.xticks(np.arange(0.0, 151.0, 50))
    if g == 0:
        plt.ylabel('$C_{NP},\ [mol/L]$', fontsize=18)
```

```
    else:
        plt.tick_params(labelleft=False)
    # plt.yscale('log')  # optional log scale
    plt.tick_params(labelsize=14)
    if g == g_num//2:
        plt.xlabel('$Time\ [s]$', fontsize=18)  # title for the X-axis
    plt.legend(loc='upper right', prop={'size': 12})

    # STEP_9B
    with open(script_folder + '/model/' + group['Group name'] + '_model.txt', 'w', encoding="utf8") as
data_output:
        data_lines = 'Time [s]' + '\t' + 'Conc [M] (mean)' + '\t' + 'SD' + '\t' + 'Conc [M] (model)' + '\n'
        for e1, e2, e3, e4 in zip(group['Mean+SD']['t'], group['Mean+SD']['Mean'], group['Mean+SD']['SD'],
group['Model']):
            data_lines += str(e1) + '\t' + str(e2) + '\t' + str(e3) + '\t' + str(e4) + '\n'
        data_output.write(data_lines)
plt.show()  # show the graph

# STEP_10
# exporting the rate constants into a .txt file
rate_constants = [gr['k_all'] for gr in samples_dict['Groups']]
with open(script_folder + '/model/' + 'Individual rate constants.txt', 'w', encoding="utf8") as rate_output:
    rate_lines = 'Sample\tReplicates (k)\n'
    for g, group in zip(rate_constants, samples_dict['Groups']):
        rate_lines += group['Group name'] + '\t'
        for r in g:
            rate_lines += str(r) + '\t'
        rate_lines += '\n'
    rate_output.write(rate_lines)
```

The source code is available at https://github.com/dot0nine/PyKinet

This script is used to fit the model by Ballauf et al. (https://doi.org/10.1021/jp101125j) to the kinetics data for 4-nitrophenol reduction with sodium borohydride catalyzed by metal nanoparticles

Guidelines for the input data for the script:

- The directory with the script should contain two subdirectories named "raw" and "model"
- The subdirectory "raw" contains files that a user wants to process
- The subdirectory "model" contains files that the script produces while processing the corresponding files from "raw"
- Names of files in "raw" should end with "_SS_raw.txt" or "_SS_raw.TXT" to be recognised by the script and get processed (SS stands for steady state).
  Also, the file name is expected to start with sX where X is one or several digits to denote the sample. Therefore, the file name should have the following format: "sXX(any characters)_SS_raw.txt"
- The corresponding file with the results of processing is saved in "model" and it has the same name as the original data file but ends with "_model.txt"
- The data files in "raw" should be formatted according to the template:
  - Block 1: parameter name and its value separated by a tab or a space
    - The user doesn't have to set the parameters in the file but it's the best way to change them. If they are not set in the file, the default values from the script are used.
    - The parameters used:
      - Knp_init – the initial (before optimization) value of the Langmuir adsorption constant of 4-nitrophenol (a fitting parameter)

- Kbh_init – the initial (before optimization) value of the Langmuir adsorption constant of sodium borohydride (a fitting parameter)
- k_init – the initial (before optimization) value of the rate constant k (a fitting parameter)
- n_init – the initial (before optimization) Langmuir−Freundlich exponent for 4-nitrophenol (a fitting parameter)
- m_init – the initial (before optimization) Langmuir−Freundlich exponent for sodium borohydride (a fitting parameter)
- Cbh – the concentration of sodium borohydride (a fixed parameter)
- S – the total surface area of nanoparticles (a fixed parameter)

- According to the model, any of these parameters can be fixed or fitted
- Don't change the names of the parameters. Change only their values.

Separator line: 20 dash ('-') symbols. It is used to divide the Block 1 from the Block 2. This line is necessary for successful parsing of the files

Block 2: the data starting with the line of headers

- Starts with one line with the headers. It doesn't matter what the headers are
- Should contain numbers in the rest of the lines. Numbers in different columns should be divided with tab or space characters.
  This is applied automatically when you simply copy-paste data from Excel.

Requirements

Executing the script requires installation of the following Python libraries: os, re, numpy, scipy, matplotlib, and lmfit.

The performance was tested only with Python 3.6.7, numpy 1.19.1, scipy 1.5.2, matplotlib 3.3.1, lmfit 1.0.1

The script works in several steps:

1) Necessary libraries are imported, the default directories are set up

2) Default values for the parameters are set up

3) Looks for data files in the subdirectory /raw, the names of the files should end with _SS.txt to get recognized and processed

4) Reads the data files line by line and extracts the data according to the pattern of data files:

a. Block 1: parameter name and its value separated by a tab or a space
b. Separator line: 20 dash symbols
c. Block 2: the data starting with the line of headers

5) Groups the data sets by samples (all replicats of the same sample belong to one group)

6) The system of ODEs and the objective function (the one that gets minimized by the solver) are described

7) For each data set fitting is set up and the optimal values of the parameters are found

8) Both input and fitted data are plotted

A. For individual samples
B. For grouped samples

9) Both input and fitted data are exported to a txt file

A. For individual samples
B. For grouped samples

10) All rate constants are exported to a txt file