Supplementary Material for:

An Investigation of the Mechanism of Formation of a Thiolate-Ligated Fe(III)-OOH.

Elaine Nam, Rodney D. Swartz, Pauline E. Alokolaro, Morgan C. Gleaves, Jessica Pikul, and Julie A. Kovacs*

I. Experimental Section.

Experimental details regarding spectrophotometric determination of pKa values for proton donors HA (HO_2, MeOH, PhOH, *p*-NO_2-PhOH, NH_4^+) in THF.

II. Derivation of half-order rate equations expressed in terms of absorbance, a measurable quantity.

III. Supplementary Figures.

Figure S-1. Residuals for the global non-linear fit (using the MATLAB program below) to the kinetic data for MeOH–induced formation of hydroperoxo $[Fe^{III}(S^{Me2}N_4(tren))(OOH)]^+$ (**2**) in THF at -78 ˚C. $[O_2^-]$= 0.19 mM, [MeOH]= 300 mM.

Figure S-2. Log(kobs) vs log([FeII]) plot for MeOH–promoted hydroperoxo $[Fe^{III}(S^{Me2}N_4(tren))(OOH)]^+$ (**2**) formation in THF at -78 ˚C. $[O_2^-]$= 0.19 mM, [MeOH]= 300 mM.

Figure S-3. kobs vs $[HA]^{1/2}$ plot for proton donor HA= *p*-CF_3-PhOH-induced $[Fe^{III}(S^{Me2}N_4(tren))(OOH)]^+$ (**2**) formation in THF at -78 ˚C. $[O_2^-]$= 0.35 mM, $[Fe^{II}]$= 6.5 mM.

Figure S-4. kobs vs $[HA]^{1/2}$ plot for proton donor HA= *p*-NH_2-PhOH-induced $[Fe^{III}(S^{Me2}N_4(tren))(OOH)]^+$ (**2**) formation in THF at -78 ˚C. $[O_2^-]$= 0.35 mM, $[Fe^{II}]$= 6.5 mM.

Figure S-5. kobs vs $[HA]^{1/2}$ plot for proton donor HA= MeOH-induced $[Fe^{III}(S^{Me2}N_4(tren))(OOH)]^+$ (**2**) formation in THF at -78 ˚C. $[O_2^-]$= 0.35 mM, $[Fe^{II}]$= 6.5 mM.

Figure S-6. kobs vs $[HA]^{1/2}$ plot for proton donor HA= MeOD-induced $[Fe^{III}(S^{Me2}N_4(tren))(OOH)]^+$ (**2**) formation in THF at -78 ˚C. $[O_2^-]$= 0.35 mM, $[Fe^{II}]$= 6.5 mM.

Figure S-7. kobs vs $[HA]^{1/2}$ plot for proton donor HA= PhOH-induced $[Fe^{III}(S^{Me2}N_4(tren))(OOH)]^+$ (**2**) formation in THF at -78 ˚C. $[O_2^-]$= 0.35 mM, $[Fe^{II}]$= 6.3 mM.

Figure S-8. kobs vs $[HA]^{1/2}$ plot for proton donor HA= *p*-I-PhOH-induced $[Fe^{III}(S^{Me2}N_4(tren))(OOH)]^+$ (**2**) formation in THF at -78 ˚C. $[O_2^-]$= 0.35 mM, $[Fe^{II}]$= 6.5 mM.

Figure S-9. kobs vs $[HA]^{1/2}$ plot for proton donor HA= *p*-Br-PhOH-induced $[Fe^{III}(S^{Me2}N_4(tren))(OOH)]^+$ (**2**) formation in THF at -78 ˚C. $[O_2^-]$= 0.35 mM, $[Fe^{II}]$= 6.7 mM.

Figure S-10. Determination of the reaction order (half) with respect to HA = *p*-NH_2-PhOH, using a log(kobs) vs log([HA]) plot, for the proton-assisted formation of $[Fe^{III}(S^{Me2}N_4(tren))(OOH)]^+$ (**2**) from $[Fe^{II}(S^{Me2}N_4(tren))]^+$ (**1**) + (18-crown-6-K+)(O_2^-) in THF at -78 ˚C. $[O_2^-]$= 0.35 mM, $[Fe^{II}]$= 6.5 mM.

Figure S-11. Determination of the reaction order (half) with respect to HA = MeOH, using a log(kobs) vs log([HA]) plot, for the proton-assisted formation of $[Fe^{III}(S^{Me2}N_4(tren))(OOH)]^+$ (**2**) from $[Fe^{II}(S^{Me2}N_4(tren))]^+$ (**1**) + (18-crown-6-K+)(O_2^-) in THF at -78 ˚C. $[O_2^-]$= 0.35 mM, $[Fe^{II}]$= 6.5 mM.

Figure S-12. Determination of the reaction order (half) with respect to HA = MeOD, using a log(kobs) vs log([HA]) plot, for the proton-assisted formation of $[Fe^{III}(S^{Me2}N_4(tren))(OOH)]^+$ (**2**) from $[Fe^{II}(S^{Me2}N_4(tren))]^+$ (**1**) + (18-crown-6-K+)(O_2^-) in THF at -78 ˚C. $[O_2^-]$= 0.35 mM, $[Fe^{II}]$= 6.5 mM.

Figure S-13. Determination of the reaction order (half) with respect to HA = p-I-PhOH, using a log(kobs) vs log([HA]) plot, for the proton-assisted formation of $[Fe^{III}(S^{Me2}N_4(tren))(OOH)]^+$ (**2**) from $[Fe^{II}(S^{Me2}N_4(tren))]^+$ (**1**) + (18-crown-6-K+)(O_2^-) in THF at -78 ˚C. $[O_2^-]$= 0.35 mM, $[Fe^{II}]$= 6.5 mM.

Figure S-14. Determination of the reaction order (half) with respect to HA = PhOH, using a log(kobs) vs log([HA]) plot, for the proton-assisted formation of $[Fe^{III}(S^{Me2}N_4(tren))(OOH)]^+$ (**2**) from $[Fe^{II}(S^{Me2}N_4(tren))]^+$ (**1**) + (18-crown-6-K+)(O_2^-) in THF at -78˚C. $[O_2^-]$= 0.35 mM, $[Fe^{II}]$= 6.5 mM.

Figure S-15. Determination of the reaction order (half) with respect to HA = p-Br-PhOH, using a log(kobs) vs log([HA]) plot, for the proton-assisted formation of $[Fe^{III}(S^{Me2}N_4(tren))(OOH)]^+$ (**2**) from $[Fe^{II}(S^{Me2}N_4(tren))]^+$ (**1**) + (18-crown-6-K+)(O_2^-) in THF at -78 ˚C. $[O_2^-]$= 0.35 mM, $[Fe^{II}]$= 6.5 mM.

Figure S-16. k_{obs} vs $[Fe^{II}]$ plot for a proton donor (HA = p-NO_2-PhOH) strong enough to completely convert O_2^- to HO_2 . Under these conditions,(i.e., when $[HO_2]$= $[O_2^-]$) the rate expression simplifies to the second order expression shown in equation (2), and the rate constant k_2 can be obtained directly from the slope.

Figure S-17. log(kobs) vs log([FeII]) plot showing first order dependence on iron for NH_4^+ promoted ($[NH_4^+]$= 0.31 mM) formation of $[Fe^{III}(S^{Me2}N_4(tren))(OOH)]^+$ (**2**) from $[Fe^{II}(S^{Me2}N_4(tren))]^+$ (**1**) + (18-crown-6-K+)(O_2^-) in THF at -78 ˚C. Under these conditions O_2^- is converted to HO_2.

Figure S-18. kobs vs [p-NO2-PhOH] plot showing zero order dependence on proton donor when $[Fe^{III}(S^{Me2}N_4(tren))(OOH)]^+$ (**2**) formation occurs in the presence of acids strong enough to convert O_2^- to HO_2.

Figure S-19. kobs vs $[NH_4^+]$ plot showing zero order dependence on proton donor when $[Fe^{III}(S^{Me2}N_4(tren))(OOH)]^+$ (**2**) formation occurs in the presence of acids strong enough to convert O_2^- to HO_2.

Figure S-20. Kinetics of methanol-induced conversion of $[Fe^{III}(S^{Me2}N_4(tren))(OOH)]^+$ (**2**) to $[Fe^{III}(S^{Me2}N_4(tren))(OMe)]^+$ (**3**) in MeOH at –78 ˚C.

Figure S-21. Possible reaction pathways resulting from protonation of hydroperoxo $[Fe^{III}(S^{Me2}N_4(tren))(OOH)]^+$ (**2**) at either the proximal or distal peroxo oxygen.

Figure S-22. Representative absorbance vs wavelength plot for the kinetics of HOAc-induced conversion of $[Fe^{III}(S^{Me2}N_4(tren))(OOH)]^+$ (**2**) to $[Fe^{III}(S^{Me2}N_4(tren))(O(H)Me)]^+$ (**4**) in MeOH at –78 ˚C.

Figure 23. Eyring plot for AcOH-induced release of H_2O_2 from hydroperoxo $[Fe^{III}(S^{Me2}N_4(tren))(OOH)]^+$ (**2**) to afford methanol-bound $[Fe^{III}(S^{Me2}N_4(tren))(O(H)Me)]^+$ (**4**).

Figure S-24. Kinetics of deuterated acetic acid-induced release of H_2O_2 from $[Fe^{III}(S^{Me2}N_4(tren))(OOH)]^+$ (**2**) to afford $[Fe^{III}(S^{Me2}N_4(tren))(O(H)Me)]^+$ (**4**) in MeOH at –78 ˚C.

IV. Source code for the MATLAB program, “KinCalc”, written by Rodney Swartz, to do the global non-linear fits to our kinetic data.

I. UV-vis Spectrophotometric Determination of pKa Values for Proton Donors HA (HO_2, MeOH, PhOH, *p*-NO_2-PhOH, NH_4^+) in THF. In order to establish an acidity scale, in THF, for the proton donors used in this kinetic study the following method, based on that reported by Streitwieser and coworkers,[1] was utilized. This involved measuring equilibrium ion pair acidity constants, K (eqn (1), for the reaction between Li+ salts of highly colored indicator dyes (Ind^-), and proton donors HA in THF.

$$\mathrm{Ind^-Li^+} + \mathrm{HA} \overset{K}{\rightleftarrows} \mathrm{IndH} + \mathrm{A^-Li^+} \qquad (1)$$

The deprotonated carbanionic indicator Ind^-Li^+ (Ind^- = deprotonated fluorene (Fl; λ_{max}= 373(9300); pKa(THF) = 22.9; deprotonated triphenylmethane (TPM; λ_{max}= 488(29000); pKa(THF) = 31.26), benzanthrene (BA; λ_{max}= 447(35000); pKa(THF) = 20.13), 1,2,3,4,5 Pentaphenyl 1,3 cyclopentadiene (Cp; λ_{max}= 346(41200); pKa(THF) = 11.07) used are highly colored with established electronic absorption parameters (λ_{max}, extinction coeff) in THF.[1,2] The resulting ΔpKa values in THF (eqn (2)) are converted to absolute

$$\Delta pK = \log(K) \qquad (2)$$

$$pK(HA) = pK(IndH) + \Delta pK \qquad (3)$$

numbers (eqn (3)) by setting the pKa of fluorene in THF equal to its ionic pKa of 22.9 (per hydrogen) measured in DMSO.[2] A typical experiment involved, the equilibration of known concentrations of both indicator anion (Ind^-M^+) and HA, and measuring the decrease in absorbance, which corresponds to the amount of the indicator anion converted to both the substrate anion and protonated indicator (IndH). At equilibrium, $[HA] = [HA]_o - [Ind^-]$, $[Ind^-]$ is measured experimentally, and both $[A^-]$ and [HInd] are equal to the difference between $[\mathrm{Ind}^-]_o$ and $[\mathrm{Ind}^-]$. The deprotonated fluorene (Fl) carbanion was used to determine the pK_a of HO_2 and PhOH, that of triphenylmethane (TPM) was used to determine the pKa of MeOH, that of benzanthrene (BA) was used to determine the pK_a of *p*=NO_2-PhOH, and that of 1,2,3,4,5 Pentaphenyl 1,3 cyclopentadiene (Cp) was used to determine the pKa of NH_4^+. This procedure required high purity materials and solvent since even trace amounts of water will quench the indicator carbanions. Precautions were taken to provide rigorous elimination of moisture by baking all glassware and storing all the samples under nitrogen in a glovebox. The decrease in the absorption due to the moisture on the surface of the sample container was evaluated in control experiments and the necessary corrections were made. All solutions were prepared daily (via BuLi (1.6 M in hexane) IndH deprotonation in THF), and measurements were carried out in a Vacuum Atmospheres glovebox under an atmosphere of nitrogen. Control experiments to measure λ_{max} and extinction coefficients were performed for each for the indicator dye generated ($\mathrm{Ind}^-\mathrm{Li}^+$) and compared to published literature values.[1] Unless otherwise stated, chemical reagents purchased from commercial vendors were of the highest purity available and used without further purification. THF was dried over alumina columns from a Solvent Degassing System (SDS, stored over calcium hydride, and vacuum transferred as needed. Electronic absorption spectra were recorded using a Varian Cary 50 spectrophotometer equipped with a "C-technologies" fiber optic cable. The fiber-optic cable was connected to the glovebox with an Ocean optics 400 μm VFT series vis/nIR vacuum feedthrough, and Ocean optics solar resistant 400 μm patchcords.

(1) Streitwieser, A.; Wang, D. Z.; Stratakis, M.; Facchetti, A.; Gareyev, R.; Abbotto, A.; Krom, J. A.; Kilway, K. V. *Can. J. Chem.* **1998**, *76*, 765-769.
(2) Kaufman, M. J.; Gronert, S.; Streitwieser, A. *J. Am. Chem. Soc.* **1988**, *110*, 2829-2835.

II. Derivation of half-order rate equations expressed in terms of absorbance, a measurable quantity.

The following is a version of the equation derived for half-order reactions expressed I terms of reactant disappearance (in "Chemical Kinetics and Reaction Mechanisms", James Espenson, 2nd edition, 2002, McGraw-Hill, pg 29.), modified in terms of product growth.

We know that changes in absorbance are proportional to changes in the concentration of our hydroperoxo intermediate $[Fe^{III}(S^{Me2}N_4(tren))(OOH)]^+$ (**2**) (abbreviated "FeOOH" from hereon), and superoxide as shown in eqn. (1).

$$\frac{\delta A}{\delta t} = \varepsilon \frac{\delta[FeOOH]}{\delta t} = -\varepsilon \frac{\delta[O_2^-]}{\delta t} \tag{1}$$

Taking into account contributions to the absorbance by species other than the species of interest, we arrive at the following equation expressed in terms of fractional absorbance changes, where, $[FeOOH]_0$, the initial concentration of the hydroperoxo, is assumed to be zero, $[FeOOH]_\infty$ is the final concentration of the hydroperoxo, and $[FeOOH]_t$ is the concentration of the hydroperoxo at time t.

$$\frac{A_t - A_0}{A_\infty - A_0} = \frac{\varepsilon([FeOOH]_t - [FeOOH]_0)}{\varepsilon([FeOOH]_\infty - [FeOOH]_0)} = \frac{[FeOOH]_t}{[FeOOH]_\infty} \tag{2}$$

If we assume that all of the added superoxide converts to iron peroxo, then the concentration of our hydroperoxo relates to the concentration of superoxide as follows

$$\begin{aligned} [FeOOH]_t &= [O_2^-]_0 - [O_2^-]_t \\ [FeOOH]_0 &= [O_2^-]_\infty \\ [FeOOH]_\infty &= [O_2^-]_0 \end{aligned} \tag{3}$$

Where, $[O_2^-]_0$, is the initial concentration of the superoxide, $[O_2^-]_\infty$ is the final concentration of concentration of the superoxide, and $[O_2^-]_t$ is the concentration of the superoxide at time t. We can relate the fractional change in peroxo concentration to that of superoxide as shown in eqn. (4)

$$\frac{[FeOOH]_t}{[FeOOH]_\infty} = \frac{[O_2^-]_0 - [O_2^-]_t}{[O_2^-]_0} \tag{4}$$

Combining eqn (2) with eqn (4) gives us the proper relationship between the fractional change in absorbance and the fractional change in superoxide concentration (eqn (5))

$$\frac{A_t - A_0}{A_\infty - A_0} = 1 - \frac{[O_2^-]_t}{[O_2^-]_0} \tag{5}$$

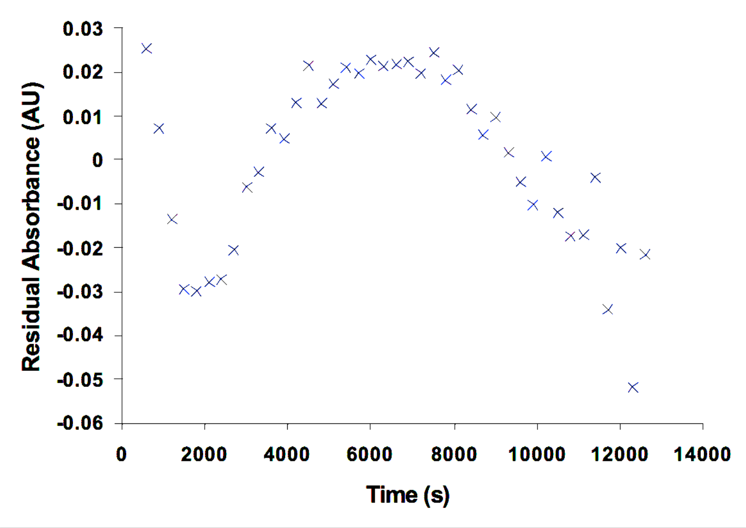
From the integrated rate expression for a half-order reaction, we know

$$\left(\frac{[O_2^-]_t}{[O_2^-]_0}\right)^{1/2} = \frac{k_{obs}t}{2[O_2^-]_0^{1/2}} + 1 \tag{6}$$

Rearranging and combining equations (5) and (6), we get

$$\left(1 - \frac{A_t - A_0}{A_\infty - A_0}\right)^{1/2} = \frac{k_{obs}t}{2[O_2^-]_0^{1/2}} + 1 \tag{7}$$

Supplemental Figures



Time	Residual
600	0.025398
900	0.0071903
1200	-0.013595
1500	-0.029339
1800	-0.029813
2100	-0.027767
2400	-0.027203
2700	-0.020455
3000	-0.0061526
3300	-0.0029168
3600	0.0071168
3900	0.0046999
4200	0.012978
4500	0.021433
4800	0.012871
5100	0.017192
5400	0.021008
5700	0.019677
6000	0.022952
6300	0.021315
6600	0.021756
6900	0.022367
7200	0.019761
7500	0.024461
7800	0.018269
8100	0.020364
8400	0.011406
8700	0.0057401
9000	0.0097507
9300	0.0017464

9600	-0.0049863
9900	-0.010199
10200	0.00068934
10500	-0.011885
10800	-0.017436
11100	-0.017192
11400	-0.0039271
11700	-0.033992
12000	-0.020107
12300	-0.051637
12600	-0.021538

Figure S-1. Residuals for the global non-linear fit (using the MATLAB program below) to the kinetic data for MeOH–induced formation of hydroperoxo $[Fe^{III}(S^{Me2}N_4(tren))(OOH)]^+$ (**2**) in THF at -78 ˚C. $[O_2^-]$= 0.19 mM, [MeOH]= 300 mM.

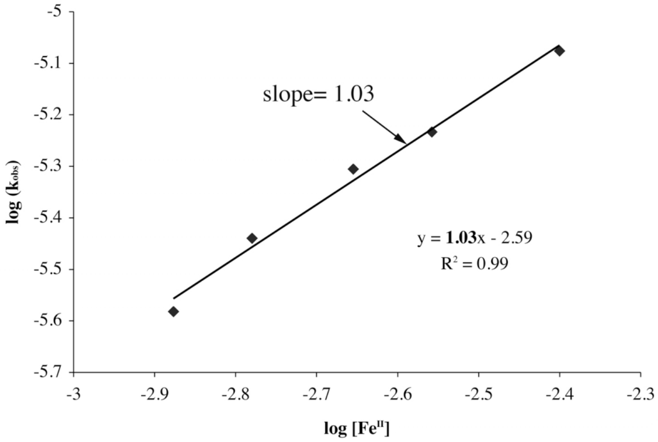


Figure S-2. Log(kobs) vs log([FeII]) plot for MeOH–promoted hydroperoxo $[Fe^{III}(S^{Me2}N_4(tren))(OOH)]^+$ (**2**) formation in THF at -78 ˚C. $[O_2^-]$= 0.19 mM, [MeOH]= 300 mM.

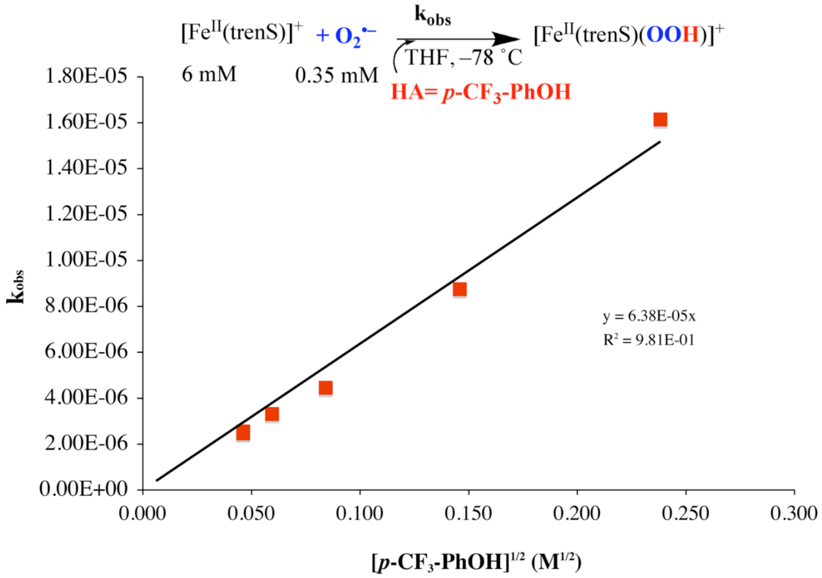


Figure S-3. kobs vs $[HA]^{1/2}$ plot for proton donor HA= *p*-CF_3-PhOH-induced $[Fe^{III}(S^{Me2}N_4(tren))(OOH)]^+$ (**2**) formation in THF at -78 °C. $[O_2^-]$= 0.35 mM, $[Fe^{II}]$= 6.5 mM.

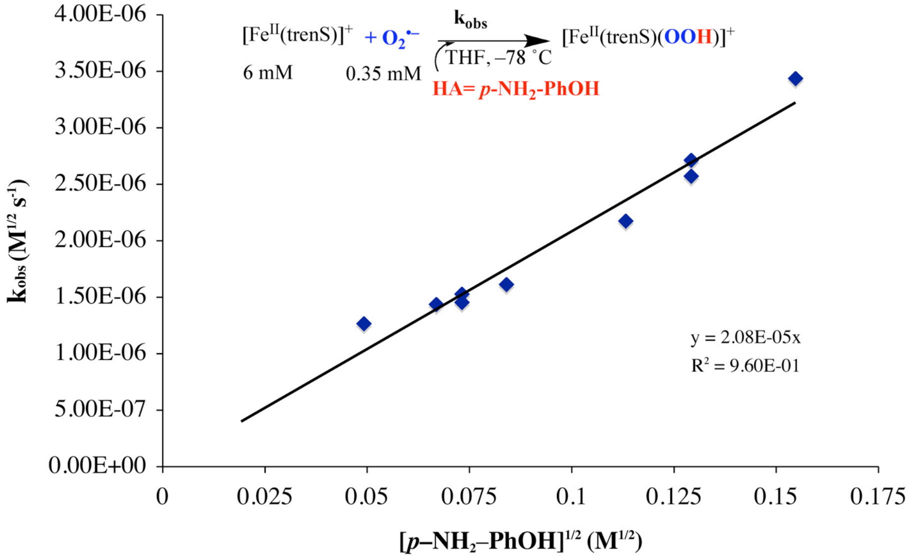


Figure S-4. kobs vs $[HA]^{1/2}$ plot for proton donor HA= *p*-NH_2-PhOH-induced $[Fe^{III}(S^{Me2}N_4(tren))(OOH)]^+$ (**2**) formation in THF at -78 °C. $[O_2^-]$= 0.35 mM, $[Fe^{II}]$= 6.5 mM.

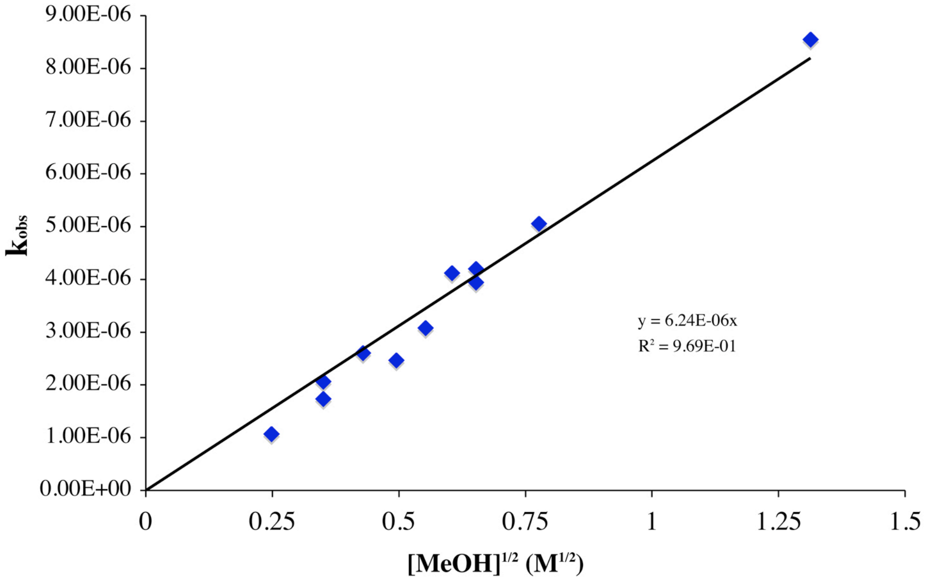


Figure S-5. kobs vs $[HA]^{1/2}$ plot for proton donor HA= MeOH-induced $[Fe^{III}(S^{Me2}N_4(tren))(OOH)]^+$ (**2**) formation in THF at -78 ˚C. $[O_2^-]$= 0.35 mM, $[Fe^{II}]$= 6.5 mM.

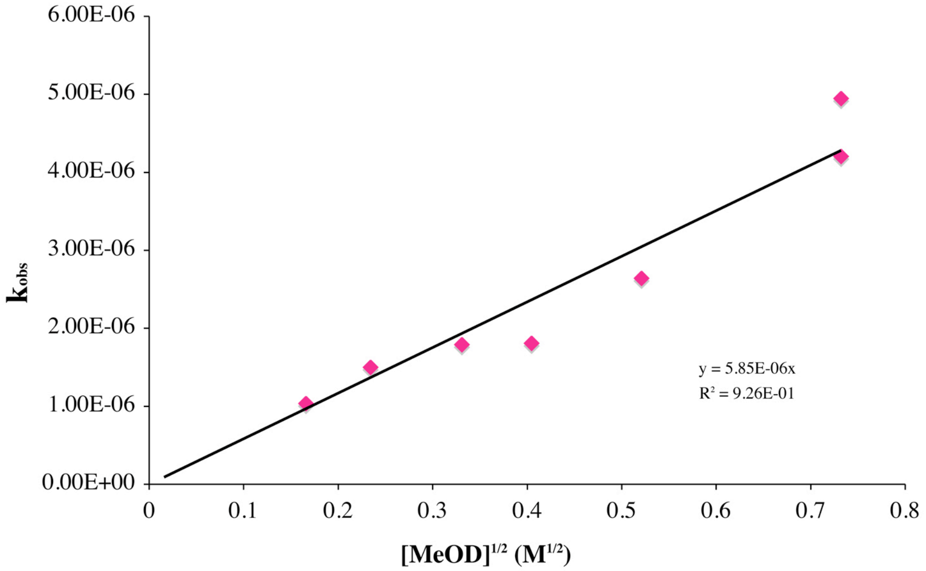


Figure S-6. kobs vs $[HA]^{1/2}$ plot for proton donor HA= MeOD-induced $[Fe^{III}(S^{Me2}N_4(tren))(OOH)]^+$ (**2**) formation in THF at -78 ˚C. $[O_2^-]$= 0.35 mM, $[Fe^{II}]$= 6.5 mM.

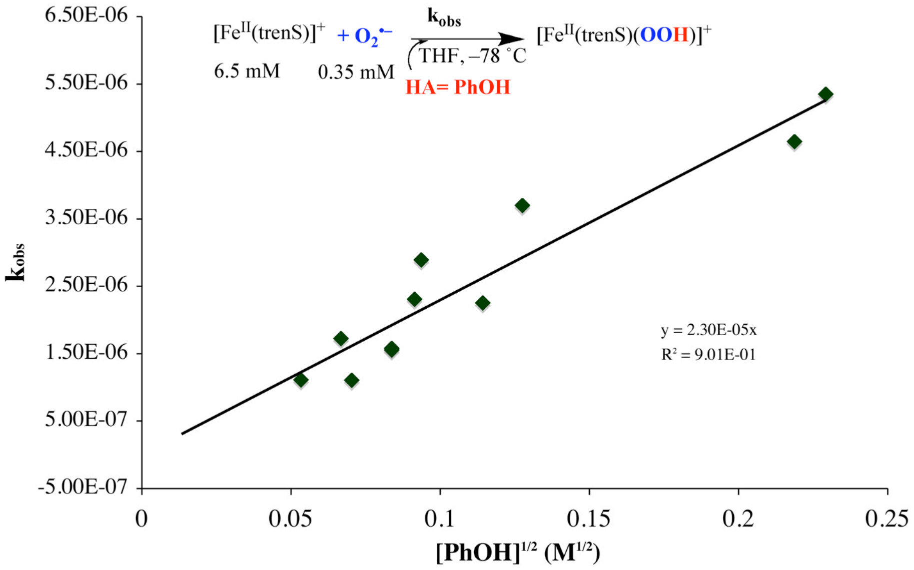


Figure S-7. kobs vs $[HA]^{1/2}$ plot for proton donor HA= PhOH-induced $[Fe^{III}(S^{Me2}N_4(tren))(OOH)]^+$ (**2**) formation in THF at -78 °C. $[O_2^-]$= 0.35 mM, $[Fe^{II}]$= 6.3 mM.

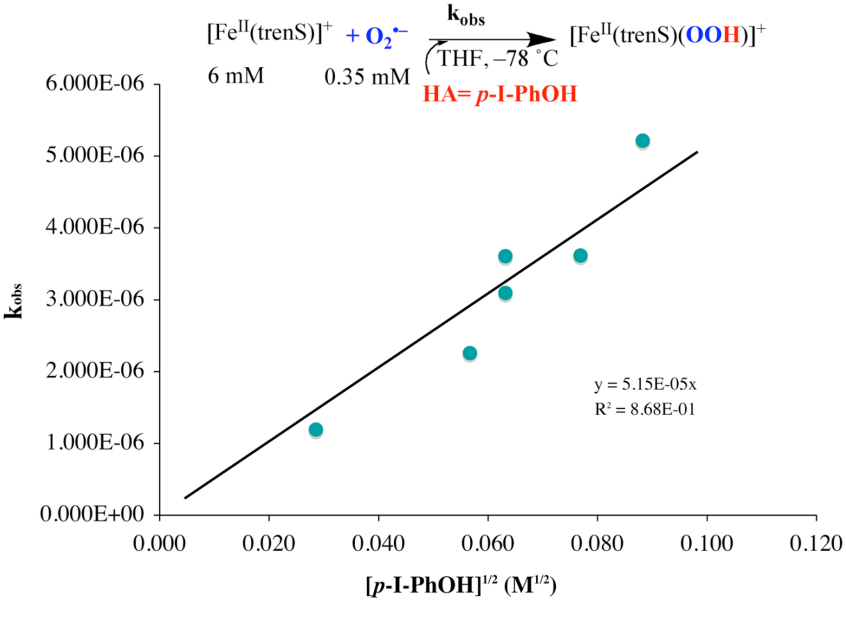


Figure S-8. kobs vs $[HA]^{1/2}$ plot for proton donor HA= *p*-I-PhOH-induced $[Fe^{III}(S^{Me2}N_4(tren))(OOH)]^+$ (**2**) formation in THF at -78 °C. $[O_2^-]$= 0.35 mM, $[Fe^{II}]$= 6.5 mM.

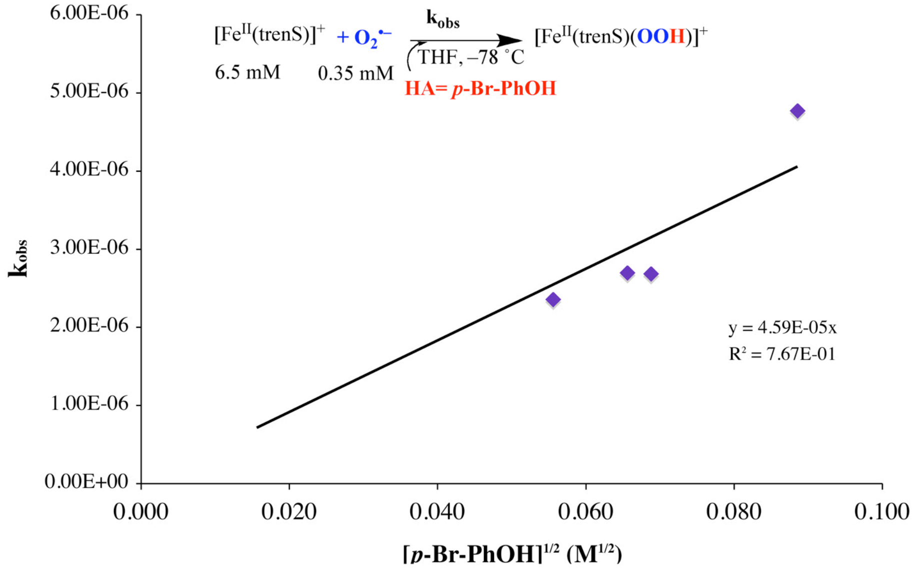


Figure S-9. kobs vs $[HA]^{1/2}$ plot for proton donor HA= *p*-Br-PhOH-induced $[Fe^{III}(S^{Me2}N_4(tren))(OOH)]^+$ (**2**) formation in THF at -78 ˚C. $[O_2^-]$= 0.35 mM, $[Fe^{II}]$= 6.7 mM.

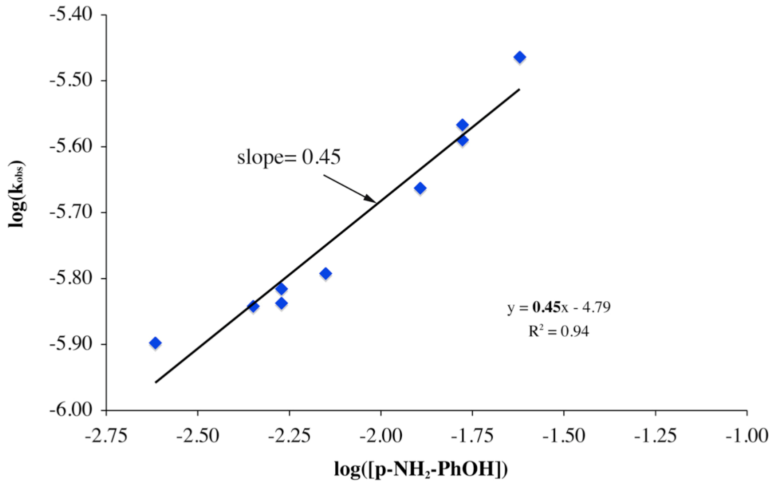


Figure S-10. Determination of the reaction order (half) with respect to HA = *p*-NH_2-PhOH, using a log(kobs) vs log([HA]) plot, for the proton-assisted formation of $[Fe^{III}(S^{Me2}N_4(tren))(OOH)]^+$ (**2**) from $[Fe^{II}(S^{Me2}N_4(tren))]^+$ (**1**) + (18-crown-6-K+)(O_2^-) in THF at -78 ˚C. $[O_2^-]$= 0.35 mM, $[Fe^{II}]$= 6.5 mM.

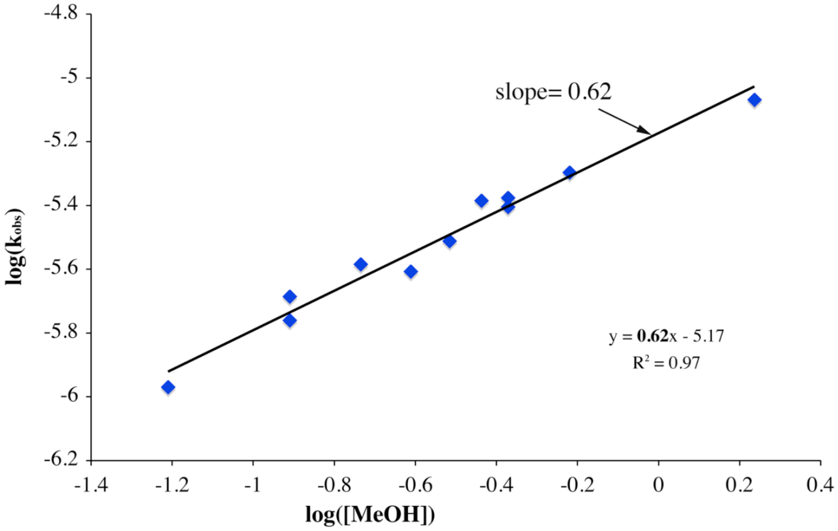


Figure S-11. Determination of the reaction order (half) with respect to HA = MeOH, using a log(kobs) vs log([HA]) plot, for the proton-assisted formation of $[Fe^{III}(S^{Me2}N_4(tren))(OOH)]^+$ (**2**) from $[Fe^{II}(S^{Me2}N_4(tren))]^+$ (**1**) + (18-crown-6-K+)(O_2^-) in THF at -78 ˚C. $[O_2^-]$= 0.35 mM, $[Fe^{II}]$= 6.5 mM.

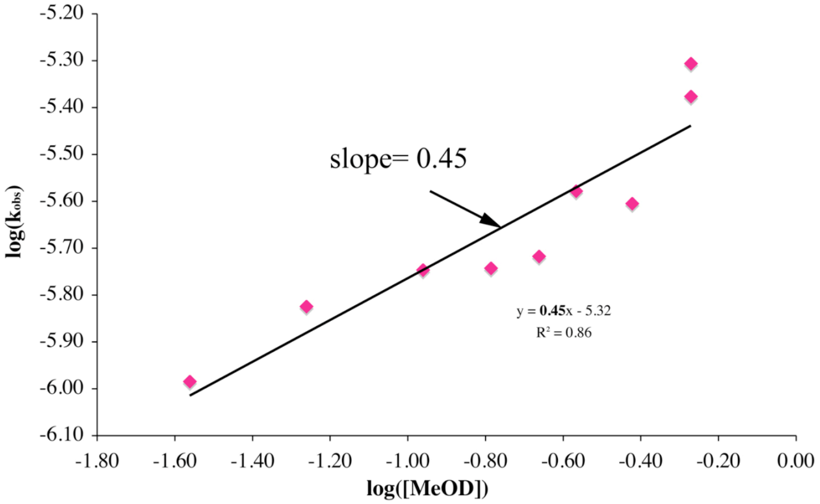


Figure S-12. Determination of the reaction order (half) with respect to HA = MeOD, using a log(kobs) vs log([HA]) plot, for the proton-assisted formation of $[Fe^{III}(S^{Me2}N_4(tren))(OOH)]^+$ (**2**) from $[Fe^{II}(S^{Me2}N_4(tren))]^+$ (**1**) + (18-crown-6-K+)(O_2^-) in THF at -78 ˚C. $[O_2^-]$= 0.35 mM, $[Fe^{II}]$= 6.5 mM.

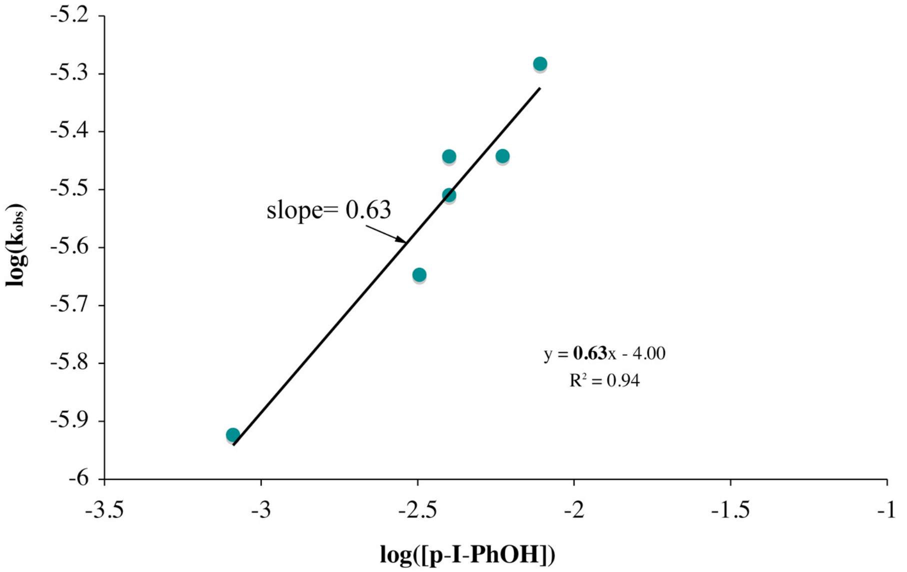


Figure S-13. Determination of the reaction order (half) with respect to HA = p-I-PhOH, using a log(kobs) vs log([HA]) plot, for the proton-assisted formation of $[Fe^{III}(S^{Me2}N_4(tren))(OOH)]^+$ (**2**) from $[Fe^{II}(S^{Me2}N_4(tren))]^+$ (**1**) + (18-crown-6-K+)(O_2^-) in THF at -78 ˚C. $[O_2^-]$= 0.35 mM, $[Fe^{II}]$= 6.5 mM.

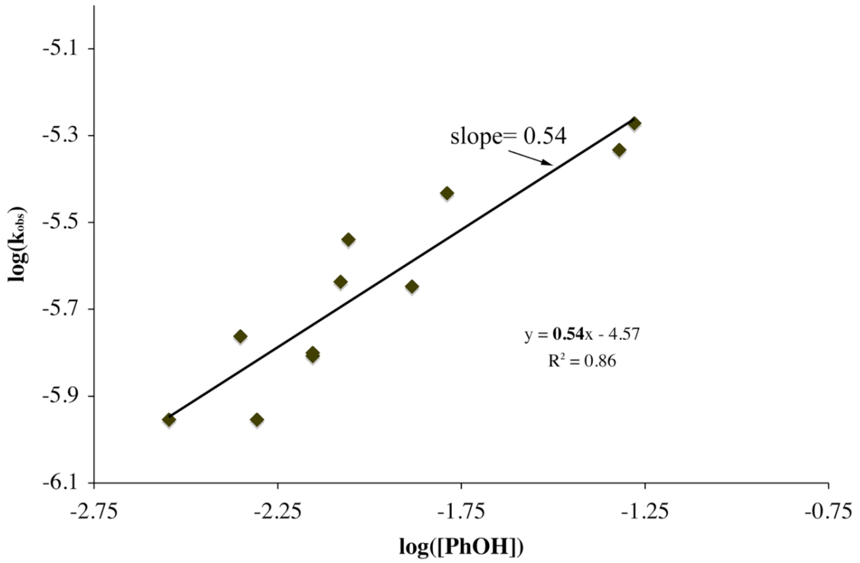


Figure S-14. Determination of the reaction order (half) with respect to HA = PhOH, using a log(kobs) vs log([HA]) plot, for the proton-assisted formation of $[Fe^{III}(S^{Me2}N_4(tren))(OOH)]^+$ (**2**) from $[Fe^{II}(S^{Me2}N_4(tren))]^+$ (**1**) + (18-crown-6-K+)(O_2^-) in THF at -78˚C. $[O_2^-]$= 0.35 mM, $[Fe^{II}]$= 6.5 mM.

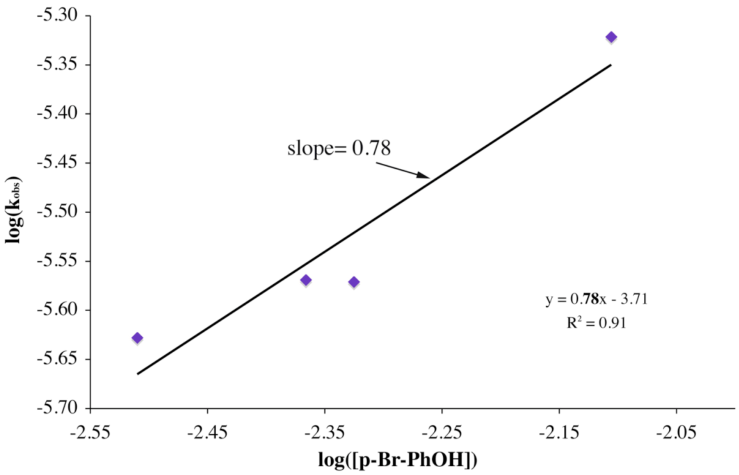


Figure S-15. Determination of the reaction order (half) with respect to HA = p-Br-PhOH, using a log(kobs) vs log([HA]) plot, for the proton-assisted formation of $[\mathrm{Fe^{III}(S^{Me2}N_4(tren))(OOH)}]^+$ (**2**) from $[\mathrm{Fe^{II}(S^{Me2}N_4(tren))}]^+$ (**1**) + (18-crown-6-K+)(O_2^-) in THF at -78 ˚C. [O_2^-]= 0.35 mM, [$\mathrm{Fe^{II}}$]= 6.5 mM.

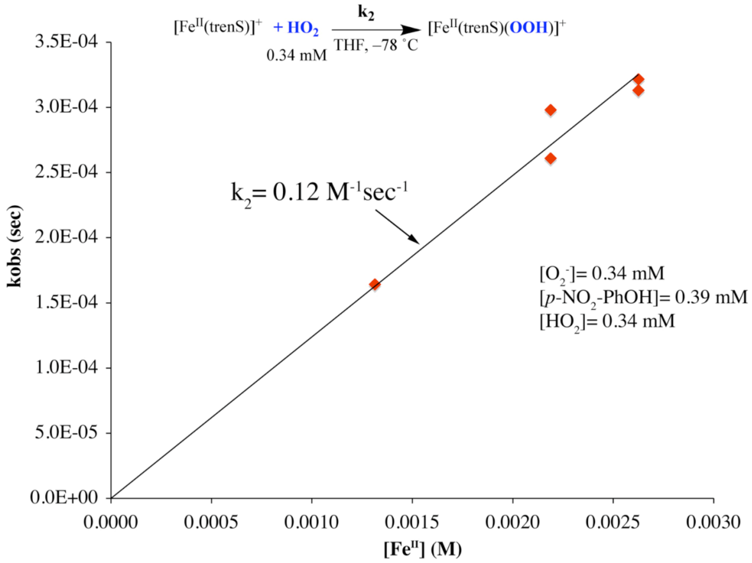


Figure S-16. k_{obs} vs [$\mathrm{Fe^{II}}$] plot for a proton donor (HA = p-NO_2-PhOH) strong enough to completely convert O_2^- to HO_2 . Under these conditions,(i.e., when [HO_2]= [O_2^-]) the rate expression simplifies to the second order expression shown in equation (2), and the rate constant k_2 can be obtained directly from the slope.

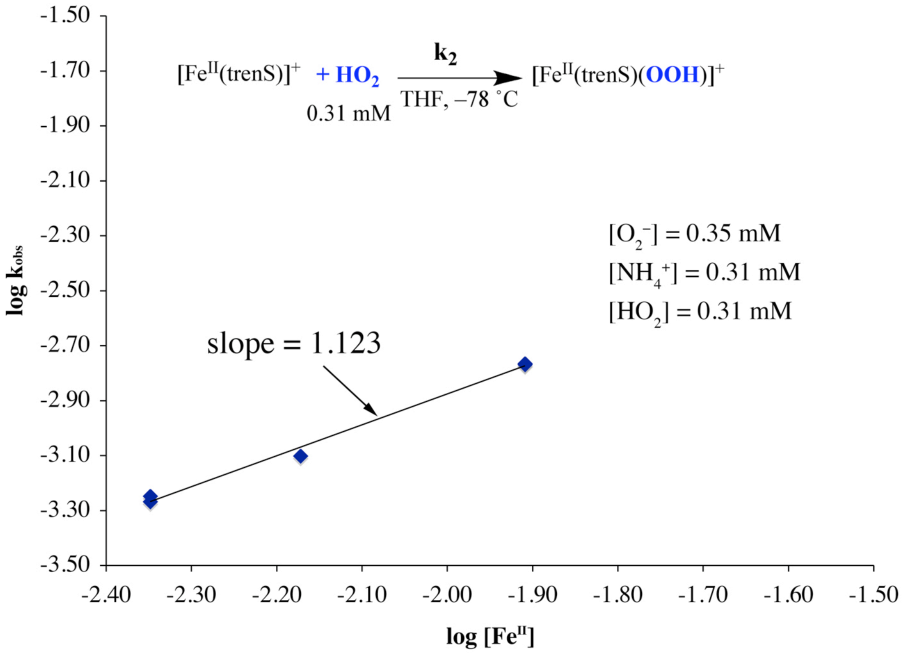


Figure S-17. log(kobs) vs log([FeII]) plot showing first order dependence on iron for NH_4^+ promoted ([NH_4^+]= 0.31 mM) formation of $[Fe^{III}(S^{Me2}N_4(tren))(OOH)]^+$ (**2**) from $[Fe^{II}(S^{Me2}N_4(tren))]^+$ (**1**) + (18-crown-6-K+)(O_2^-) in THF at -78 ˚C. Under these conditions O_2^- is converted to HO_2.

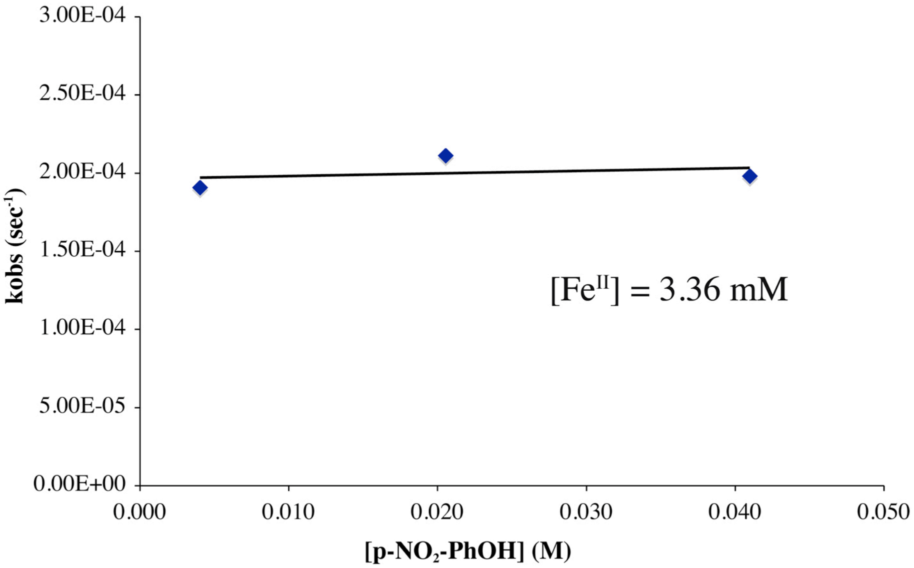


Figure S-18. kobs vs [p-NO2-PhOH] plot showing zero order dependence on proton donor when $[Fe^{III}(S^{Me2}N_4(tren))(OOH)]^+$ (**2**) formation occurs in the presence of acids strong enough to convert O_2^- to HO_2.

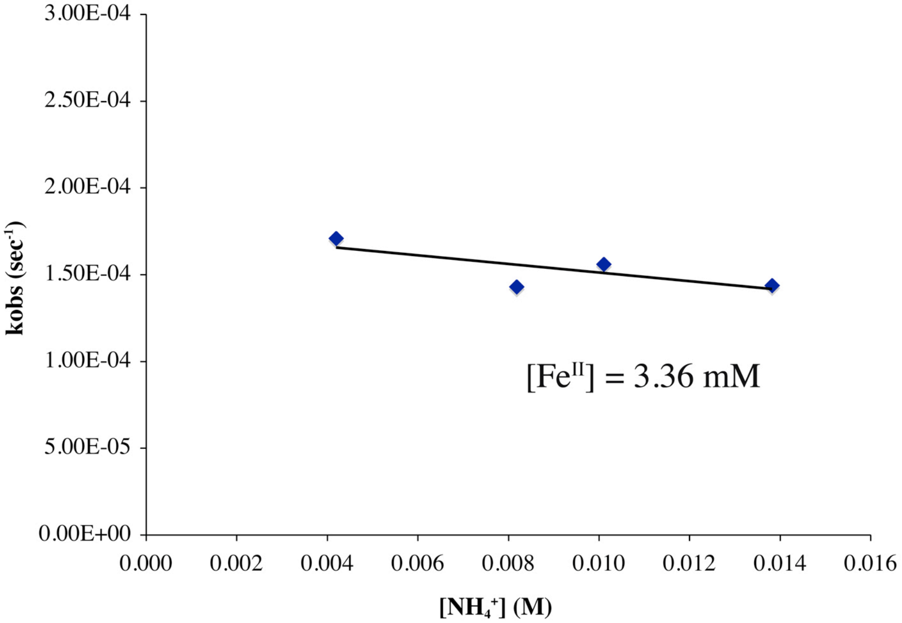


Figure S-19. kobs vs $[NH_4^+]$ plot showing zero order dependence on proton donor when $[Fe^{III}(S^{Me2}N_4(tren))(OOH)]^+$ (**2**) formation occurs in the presence of acids strong enough to convert O_2^- to HO_2.

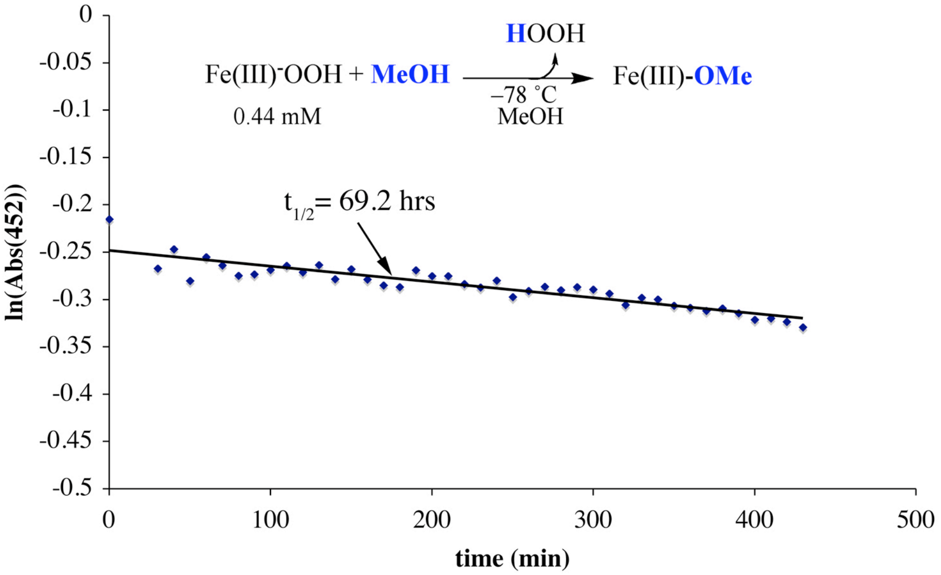


Figure S-20. Kinetics of methanol-induced conversion of $[Fe^{III}(S^{Me2}N_4(tren))(OOH)]^+$ (**2**) to $[Fe^{III}(S^{Me2}N_4(tren))(OMe)]^+$ (**3**) in MeOH at –78 ˚C.

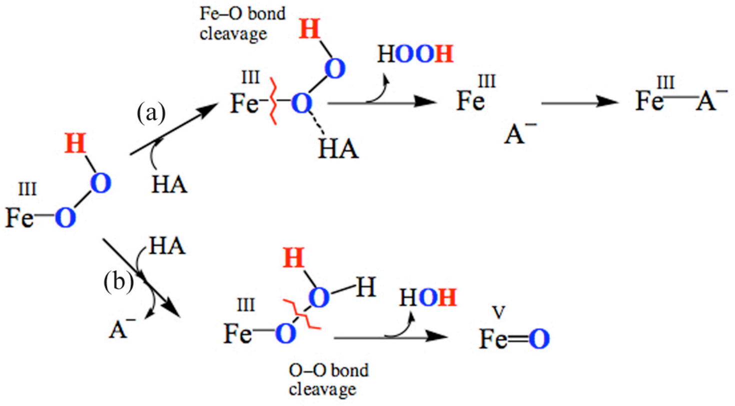


Figure S-21. Possible reaction pathways resulting from protonation of hydroperoxo $[Fe^{III}(S^{Me2}N_4(tren))(OOH)]^+$ (**2**) at either the proximal or distal peroxo oxygen.

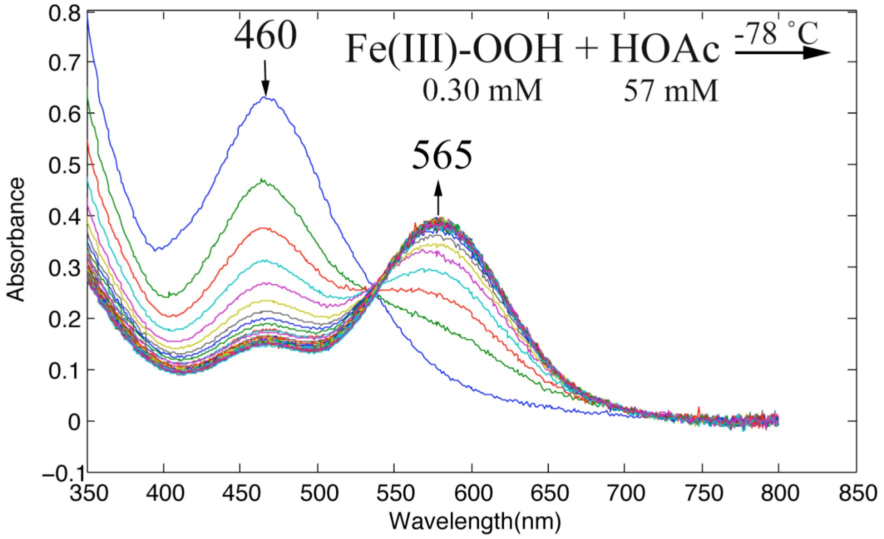


Figure S-22. Representative absorbance vs wavelength plot for the kinetics of HOAc-induced conversion of $[Fe^{III}(S^{Me2}N_4(tren))(OOH)]^+$ (**2**) to $[Fe^{III}(S^{Me2}N_4(tren))(O(H)Me)]^+$ (**4**) in MeOH at –78 °C.

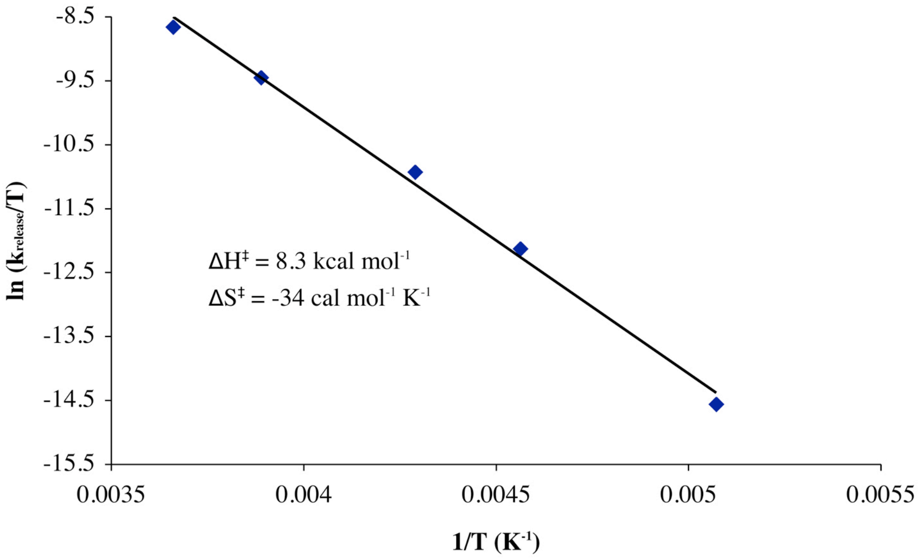


Figure 23. Eyring plot for AcOH-induced release of H_2O_2 from hydroperoxo $[Fe^{III}(S^{Me2}N_4(tren))(OOH)]^+$ (**2**) to afford methanol-bound $[Fe^{III}(S^{Me2}N_4(tren))(O(H)Me)]^+$ (**4**).

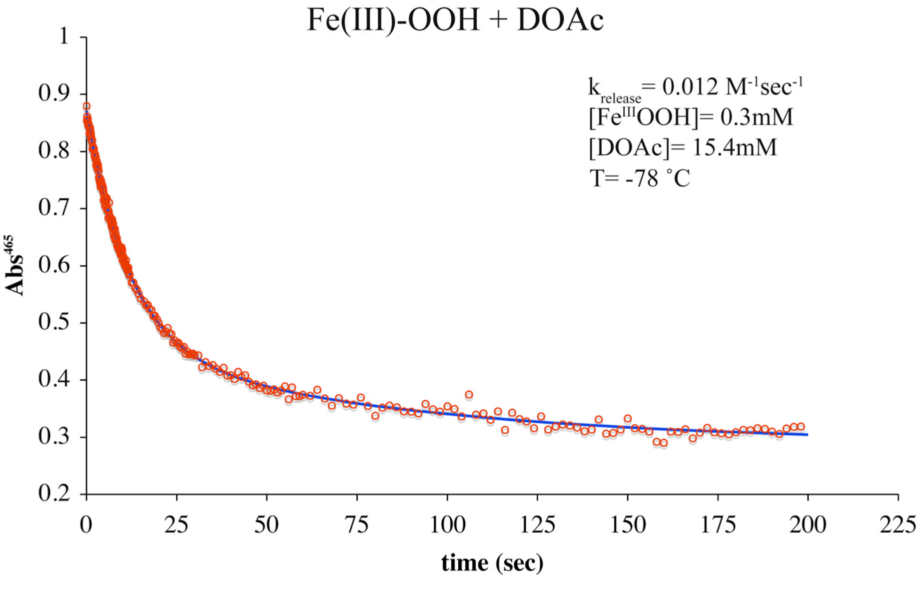


Figure S-24. Kinetics of deuterated acetic acid-induced release of H_2O_2 from $[Fe^{III}(S^{Me2}N_4(tren))(OOH)]^+$ (**2**) to afford $[Fe^{III}(S^{Me2}N_4(tren))(O(H)Me)]^+$ (**4**) in MeOH at –78 ˚C.

IV. Source code for the MATLAB program, "KinCalc", written by Rodney Swartz, to do the global non-linear fits to our kinetics data.

```
function KinCalc

% This progam determines the rate constant for various reactions
% based on initial information given
%
% Rodney Swartz
% 12/2009

try
    clear;

    switch questdlg('Do you have an initial data file?','Initial File')
        case 'Yes'
            s=getInitFromFile;
        case 'No'

            %                       %
            % Get Initial Values    %
            %                       %

            s = getInit;

        otherwise
            error('cancel')
    end

    %       %
    % Fit k %
    %       %

    s=minimizeRate(s);

    %               %
    % Write Report  %
    %               %

    s.sig_r=sqrt(s.ssq/(sum([s.ns{:}])*s.nl-length(s.par) ...
        -sum(s.absorber)*s.nl));

    s.sig_par=s.sig_r*sqrt(diag(inv(s.Curv)));

    for i=1:length(s.par)
        fprintf(1,'%s: %g +- %g\n',s.par_str{i}(3:end),s.par(i),...
            s.sig_par(i));
    end
    fprintf(1,'sig_r: %g\n',s.sig_r);

catch
    errmsg = lasterr;
    if (strfind(errmsg, 'cancel'))
        disp('Exited by User')
    else
```

```
        disp(errmsg)
    end
end
```

```
function s = getInit
%GETINIT obtains the starting values needed by KinCalc
%   s=getInit(s) obtains rate constants, ODE function, concentration
data,
%   and time data and appends them to the structure 's' for use by
KinCalc.
%   The next step is to create a function that can read all of this
%   information from a xls file.

try
    [fileName, path] = uigetfile({'*.csv', ...
        'Comma Delimited Text (*.csv)'}, 'Multiselect', 'on');
    if (~iscell(fileName))
        if(ischar(fileName))           % if one file is selected we
need to
            fileName=cellstr(fileName);% put the name into a cell array
        end
        if(isnumeric(fileName))        % if canceled is hit
            error('cancel')
        end
    end
catch
    rethrow(lasterror)
end

s.fileName = fileName;
s.path = path;
s.nm = length(fileName);

%                           %
% Estimated rate constants  %
%                           %

numberOfSteps = inputdlg('How many steps to input, including reverse');
if(isempty(numberOfSteps))
    error('cancel');
end
s.nk = str2num(numberOfSteps{1});
if(~isnumeric(s.nk))
    error('Must be an Integer')
end
s.nk=uint32(s.nk);

for i=1:s.nk
    rateDlg{i} = horzcat('Estimated rate of step ',num2str(i));
end
rateValues = inputdlg(rateDlg);
if(isempty(rateValues))
    error('cancel')
end
for i=1:length(rateValues)
    s.k(i) = str2num(rateValues{i});
    if(~isnumeric(s.k(i)))
        error('Rate must be a number');
    end
end
```

```
%                           %
% Get Function for Rcalc    %
%                           %

s.fName = uigetfile({'*.m','M-files (*.m)'},'function for RCalc');
if(isnumeric(s.fName))
    error('cancel')
end

extPlace=findstr(s.fName,'.m');
s.fName = s.fName(1:extPlace-1);
numberOfConc = inputdlg('How many concentrations to input',...
    'Number Concentration');
if(isempty(numberOfConc))
    error('cancel')
end
s.nc = str2num(numberOfConc{1});
if(~isnumeric(s.nc))
    error('Must be an Integer')
end
s.nc = uint32(s.nc);

for i=1:s.nm

    %                       %
    % Get Concentraion Data %
    %                       %

    for j=1:s.nc
        concDlg{j} = horzcat('Concentraion of
y(',horzcat(num2str(j),...
            horzcat(') for ', num2str(s.fileName{i}))));
    end
    concValues = inputdlg(concDlg);
    if(isempty(concValues))
        error('cancel')
    end
    for j=1:length(concValues)
        if(isempty(concValues{j}))
            concValues{j} = '0';
        end
        s.C_0{i}(j) = str2num(concValues{j});
        if(~isnumeric(s.C_0{i}(j)))
            error('Concentration must be a number');
        end
    end

    dlgQuestion = horzcat('Time Interval for ', s.fileName{i});
    interval = inputdlg(dlgQuestion,'Interval');
    if(isempty(interval))
        error('cancel')
    end
    s.interval(i) = str2num(interval{1});
    if(~isnumeric(s.interval(i)))
        error('Interval must be a number');
    end
end
```

```
%                                    %
% Determine which species Absorb    %
%                                    %

for i=1:s.nc
    whichConc{i} = horzcat('species ',horzcat(num2str(i),' absorb?'));
end
doesAbsorb=inputdlg(whichConc,'Which Species Absorb');
if(isempty(doesAbsorb))
    error('cancel')
end
for i=1:length(doesAbsorb)
    if(isempty(doesAbsorb{i}))
        doesAbsorb{i} = 'n';
    end
    if(doesAbsorb{i} == '1' || doesAbsorb{i}(1) == 'y' ...
            || doesAbsorb{i}(1) == 'Y')
        s.absorber(i) = 1;
    else
        s.absorber(i) = 0;
    end
end

switch questdlg('Do you want to save these values?','Save Input')
    case 'Yes'
        writeXLS(s);
end

s=compileInit(s);
```

```
function s=getInitFromFile
%GETINITFROMFILE reads initial information from an excel file
%   s=getInitFromFile reads the initial values from an excel file with
a
%   predefined organization to help streamline the calculation of
global
%   data.  This file contains a lot of "magic" numbers, their use is
often
%   to keep the indices correct.

[xlsFile s.path]=uigetfile({'*.xls','Excel Spread Sheet (*.xls)'});
if (isnumeric(xlsFile))
    error('cancel')
end

[dummyN dummyT rawdata]=xlsread(horzcat(s.path,xlsFile));

sizeOfRaw = size(rawdata);

s.fName = rawdata{2,1};
i=2;
while(i <= sizeOfRaw(2) && ~isnan(rawdata{2,i}))
    s.k(i-1)=rawdata{2,i};
    i=i+1;
end
s.nk=length(s.k);
for j=5:sizeOfRaw(1)
    s.fileName{j-4}=rawdata{j,1};
    s.interval(j-4)=rawdata{j,2};
    i=3;
    while(i<=sizeOfRaw(2) && ~isnan(rawdata{j,i}))
        s.C_0{j-4}(i-2)=rawdata{j,i};
        if(rawdata{3,i} == '1' || rawdata{3,i} == 'y' ...
                || rawdata{3,i} == 'Y')
            s.absorber(i-2) = 1;
        else
            s.absorber(i-2) = 0;
        end
        i=i+1;
    end
    s.nc=i-3;
end

s.nm=length(s.fileName);
s=compileInit(s);
```

```
function s=compileInit(s)

%                %
% Convert Data  %
%                %

try
    for i=1:s.nm
        [s.lam_temp{i}, s.Y{i}] =
convertData(horzcat(s.path,s.fileName{i}));
    end
catch
    rethrow(lasterror)
end

for i=1:s.nk
    s.par_str{i}=horzcat('s.k(',horzcat(num2str(i),')'));
end
s.par=get_par(s);

%                             %
% Check and Fix Wavelengths %
%                             %

while(true)
    try
        ts = equalizeLam(s);
    catch
        errmsg=lasterr;
        if(strfind(errmsg, 'range'))
            switch questdlg('Incorrect Wavelength Range Entered.
Continue?',...
                    'Ragne Error')
                case 'Yes'
                    continue
                otherwise
                    rethrow(lasterror)
            end
        else
            rethrow(lasterror)
        end
    end

    plot(ts.lam,ts.Y{1});
    drawnow;

    switch questdlg('Is this wavelength range acceptable?','Wavelength
Range')
        case 'Yes'
            s=ts;
            clear ts;
            break
        case 'No'
            continue
        otherwise
            error('cancel')
    end
```

```
end

%                   %
% Prepare Time Data %
%                   %

for i=1:s.nm
    plot(s.lam,s.Y{i});
    drawnow;

    sizeAbs = size(s.Y{i});
    s.ns{i} = sizeAbs(1);
    s.t{i} = 0:s.interval(i):s.interval(i)*(s.ns{i} - 1);
end

s.nl = length(s.lam);
```

```
function withZero = zero(noZero)
sizeofData = size(noZero);
i = 1;
while(i <= sizeofData(1))
    j = 1;
    data = noZero(i,1:10);
    subtracFactor = mean(data);
    while(j <= sizeofData(2))
        withZero(i,j) = noZero(i,j) - subtracFactor;
        j = j+1;
    end
    i = i+1;
end
```

```
function [lam, Y] = convertData(csvFile)
%CONVERTDATA convert file to useable form.
%   [lam Y]=convertData(csvFile) reads the raw csv file and extracts
out
%   which columns are wavelengths and which are absorption data.

try
    rawData = csvread(csvFile);
catch
    try
        errmsg = lasterr;
        if (strfind(errmsg, 'Mismatch'))
            rawData = csvread(csvFile,2,0);
        end
    catch
        errmsg = lasterr;
        if (strfind(errmsg, 'Mismatch'))
            rowLength = getRow(errmsg);
            colLength = getColLength(csvFile,2,0);
            rawData = csvread(csvFile,2,0,[2 0 rowLength colLength]);
        end
    end
end

dataSize = size(rawData);

% Know rawData(1,1) is a lam checking to see if the csv is a
% lam:Y:lam:Y pattern or a lam:Y:Y:Y pattern

if (rawData(1,1) == rawData(1,3))
    stepSize = 2;
else
    stepSize = 1;
end

lam = rawData(:,1);
Y = rawData(:, 2:stepSize:dataSize(2))';

i = 1;

if (i ~= 1)
    Y = Y(i:dataSize(2),:);
end

Y = zero(Y);

plot(lam, Y);
drawnow;
```

```
function s = equalizeLam(s)
%EQUALIZELAM determines proper wavelength range for multipe files
%   s=equalizeLam(s) reads the wavelengths and absorbances from
multiple
%   files and tries to determine what wavelength range is common
amongst
%   all files.  The program then query's the user to determine what
subset
%   of that wavelength range is desired.  This program requires that
the
%   scan rate be the same amongst all files.

maxLam = 1e9;
minLam = 0;
for i=1:s.nm
    if (maxLam > s.lam_temp{i}(1))
        maxLam = s.lam_temp{i}(1);
    end
    if (minLam < s.lam_temp{i}(length(s.lam_temp{i})))
        minLam = s.lam_temp{i}(length(s.lam_temp{i}));
    end
end

startLamDlg = horzcat('Start Wavelength (< ',
horzcat(num2str(uint32(maxLam)),')'));
endLamDlg = horzcat('End Wavelength (> ',
horzcat(num2str(uint32(minLam)),')'));

lamRange = inputdlg({startLamDlg, endLamDlg}, ...
    'Wavelength Range');
if(isempty(lamRange))
    error('cancel')
end
for i=1:length(lamRange)
    if(isempty(lamRange{i}))
        lamRange{i} = '0';
    end
end

lamStart = str2num(lamRange{2});
lamEnd = str2num(lamRange{1});

if (isempty(lamStart) || isempty(lamEnd) || ...
        ~isnumeric(lamStart) || ~isnumeric(lamEnd))
    error('Invalid range');
elseif (lamEnd > lamStart)
    lamTemp = lamStart;
    lamStart = lamEnd;
    lamEnd = lamTemp;
elseif (lamStart == lamEnd)
    error('Wavelength range is zero')
end

for i=1:s.nm

    if (lamStart > uint32(s.lam_temp{i}(1)))
        error('Wavelength start point is outside of measured range')
```

```
    end

    j = 1;
    while (uint32(s.lam_temp{i}(j)) > lamStart && j <
length(s.lam_temp{i}))
        j = j+1;
    end
    lamStartPoint = j;

    if (lamStartPoint == length(s.lam_temp{i}))
        error('Wavelength start point is outside of measured range')
    end

    if (lamEnd < uint32(s.lam_temp{i}(length(s.lam_temp{i}))))
        error('Wavelength end point is outside of measured range')
    end

    while (lamEnd < uint32(s.lam_temp{i}(j)) && j <
length(s.lam_temp{i}))
        j = j+1;
    end
    lamEndPoint = j;

    s.lam_temp{i} = s.lam_temp{i}(lamStartPoint:lamEndPoint);
    s.Y{i} = s.Y{i}(:,lamStartPoint:lamEndPoint);
end

for i=1:s.nm
    if (length(s.lam_temp{i}) ~= length(s.lam_temp{1}))
        error('Lambda Mismatch')
    end
end

s.lam = s.lam_temp{1};
```

```
function par=get_par(s)

for i=1:length(s.par_str)
    par(i,1)=eval(s.par_str{i});
end
```

```
function colLength = getColLength(file,r,c)

[fid message] = fopen(file);
if fid < 0
    error('MATLAB:dlmread:FileNotOpened',...
        'The file ''%s'' could not be opened because: 
%s',filename,message);
end

delimiter = (',');
nrows = 1;

delimiter = sprintf(delimiter);
whitespace  = setdiff(sprintf(' \b\t'),delimiter);
cresult  = textscan(fid,'',nrows,...
               'delimiter',delimiter,'whitespace',whitespace, ...
               'headerlines',r,'headercolumns',c,...
               'returnonerror',0,'emptyvalue',0);

colLength = length(cresult)-1;

fclose(fid);
```

```
function row = getRow(s)

% this function reads the error message from csvread and trys to
% find the row that caused the error

rStart = strfind(s, 'row ')+4;
rEnd = strfind(s,', field')-1;
row = str2num(s(rStart:rEnd));
```

```
function out = greatestAbsChange(A)

sizeofData = size(A);
absChange = abs(A(1,:) - A(sizeofData(1),:));
[maxAbsChange,out] = max(absChange);
```

```
function s=minimizeRate(s)
%MINIMIZERATE determines the rate constant that gives the best fit
%   s=minimizeRate(s) utilizes fminsearch and Rcalc to find the rate
%   constant that gives the best fit to the data.  A Curv matrix is
%   calculated for use in the estimation of the error in the rate
constant

[s.par, s.ssq] = fminsearch('Rcalc', s.k,[],s);
s=put_par(s);

%                                        %
% Estemation of error in rate constant  %
%                                        %

[ssq, r0]=feval('Rcalc',s);
r0 = cell2mat(r0);
delta = 1e-6;
for i=1:length(s.par)
    eval([s.par_str{i} '=' s.par_str{i} '*(1+delta);']);
    [ssq r]=feval('Rcalc',s);
    r = cell2mat(r);
    eval([s.par_str{i} '=' s.par_str{i} '/(1+delta);']);
    J(:,i)=(r-r0)/(delta*s.par(i));
end

s.Curv = J'*J;
```

```
function [ssq vargout]=Rcalc(varargin)
%RCALC error calculator used by KinCalc
%   [ssq vargout]=Rcalc(varargin) determines the error between the
%   calculated spectrum and actual spectrum, utilizing the ode solver
to
%   calculate the concentration time data and nonneg to calculate the
molar
%   absorptivites.  This program was designed for global analysis.

if nargin==1
    s=varargin{1};
end

if nargin==2
    s=varargin{2};
    s.k=varargin{1};
end

%                            %
% Obtain concentration data %
%                            %

for i=1:s.nm
    [t_dummy,s.C{i}]=ode15s(s.fName,s.t{i},s.C_0{i},[],s.k);
    if (length(s.t{i}) ~= length(s.C{i}))
        ssq=inf;
        r=sqrt(ssq);
        return
    end
end

%                                                  %
% Combine all concentrations for global analysis    %
%                                                  %

C_globTemp=vertcat(s.C{:});
j=1;
for i=1:length(s.absorber)
    if(s.absorber(i))
        C_glob(:,j) = C_globTemp(:,i);
        for k=1:s.nm
            s.CAbsorber{k}(:,j) = s.C{k}(:,i);
        end
        j=j+1;
    end
end

Y_glob=vertcat(s.Y{:});

%                                       %
% Obtain molar abs. and calculate error %
%                                       %
s.A=nonneg(Y_glob',C_glob')';
R=Y_glob-C_glob*s.A;

r=R(:);
```

```
subplot(3,2,1);

plot(s.lam,R);
xlabel('wavelength');ylabel('residual');
drawnow;

ssq=norm(r,2);

%                                       %
% Display Data and assign output args.  %
%                                       %

lamMax = greatestAbsChange(s.Y{1});

if nargout==1
    subplot(3,2,2);
    plot(s.t{1},s.C{1});axis tight;
    xlabel('time');ylabel('conc.');
    subplot(3,2,3);
    for j=1:s.nm
        plot(s.t{j},s.CAbsorber{j}*s.A(:,lamMax),s.t{j},...
            s.Y{j}(:,lamMax),'.');axis tight; hold on;
    end
    legend(cellstr(int2str([s.lam(lamMax)]')))
    xlabel('time');ylabel('absorbance'); hold off;
end
subplot(3,2,5)
plot(s.lam,s.A);axis tight;
xlabel('wavelength');ylabel('absorptivity');
drawnow;
if nargout==2
    vargout(1) = {r};
end
```

```
function [F,SS]=nonneg1(X,M,F,Options);
%function [F,SS]=nonneg1(X,M,F,Options);
%
% 'nonneg.m'
% $ Version 0.01 $ Date 6. Aug. 1997 $ Not compiled $
%
% This algorithm requires access to:
% ''
%
% Copyright
% Claus A. Andersson 1995-1997
% Chemometrics Group, Food Technology
% Department of Food and Dairy Science
% Royal Veterinary and Agricultutal University
% Rolighedsvej 30, T254
% DK-1958 Frederiksberg
% Denmark
% E-mail claus@andersson.dk
%
% ---------------------------------------------------------
%             Fast! Non-negativity Regression
% ---------------------------------------------------------
%
% [F,SS]=nonneg(X,M,F,Options);
% [F,SS]=nonneg(X,M);
%
% X         : Matrix of regressors.
% M         : Matrix of regressands.
% F         : The non-negativity constrained solution matrix.
% Options   : Type 'help Options' at the prompt.
%
% Given X and M this algorithm solves for the optimal
% F in a least squares sense, using that
%      X = F*M
% in the problem
%      min ||X-F*M||, s.t. F>=0, for given X and M.
%
% This version does not accept missing values.
format long
format compact
Show=0;

SS=[];

[aX bX]=size(X);
[aM bM]=size(M);
aF=aX;
bF=aM;
w=bF;

Xc=X;
W=eye(w);
itimax=100;
itomax=100;
SSimax=1e-10;
SSomax=1e-12;
SSiOld=realmax;
```

```
SSoOld=realmax;
%Initialize F
if ~exist('F')
   F=X*M'/(M*M');
   FOld=F;
   I=find(F<0);
   FOld(I)=0;
   SSiOld=sum(sum( (F-FOld).^2 ));
   F=FOld;
end;
FOld=F;
SSX=sum(sum(X.^2));
MMT=M*M';
XMT=X*M';
InvMMT=1./diag(MMT);

ito=0;
convo=0;
while ~convo,
   ito=ito+1;
   convi=0;
   iti=0;
   while ~convi,
      iti=iti+1;

      %Iterate on variables for non-negativity
      for i=1:w,
         W(i,i)=0;
         f=XMT(:,i)-F*W*MMT(:,i);
         f=InvMMT(i)*f;
         I=find(f<0);
         if ~isempty(I),
             %f(I)=0;
             f(I)=zeros(1,length(I));
         end;
         F(:,i)=f;
         W(i,i)=1;
      end;

      %Estimate error now
      SSi=sum(sum( (F-FOld).^2 ));
      if SSi < sum(sum( FOld.^2 ))*SSimax | iti>itimax,
         convi=1;
      end;
      FOld=F;

   end;

   %Estimate error on the transformed LS problem
   SSo=sum(sum( (XMT-F*MMT).^2 ));
   if (SSoOld-SSo)/SSX<SSomax,
      convo=1;
   end;
   if ito>itomax,
      convo=1;
   end;
   SSoOld=SSo;
```

```
end;
format
```

```
function s=put_par(s)

for i=1:length(s.par_str)
    eval([s.par_str{i} '= s.par(i);']);
end
```

```
function writeXLS(s)

[xlsFile path]=uiputfile({'*.xls','Excel Spread Sheet (*.xls)'});
data{1,1} = 'FUNCTION NAME';
data{1,2} = 'RATE';
data{2,1} = s.fName;
for i=1:s.nk
    data{2,i+1}=s.k(i);
end
data{3,2} = 'Absorb?';
for i=1:s.nc
    if s.absorber(i)
        data{3,i+2} = 'y';
    else
        data{3,i+2} = 'n';
    end
end
data{4,1} = 'FILENAME';
data{4,2} = 'INTERVAL';
for i=1:s.nc
    data{4,i+2} = horzcat('y(',horzcat(num2str(i),')'));
end
for j=1:s.nm
    data{j+4,1} = s.fileName{j};
    data{j+4,2} = s.interval;
    for i=1:s.nc
        data{j+4,i+2} = s.C_0{j}(i);
    end
end
xlswrite(horzcat(path,xlsFile),data);
```

```
function dy = peroxoRx(t,y,flag,k)
% the reaction of interest:
% HA + O2- <---> HO2 + A-
% HO2 + Fe(II) ---> Fe(III)OOH
% y(1) = HA; y(2) = O2-; y(3) = HO2; y(4) = A-; y(5) = Fe(II);
% y(6) = Fe(III)OOH;
% dHA = k3[HO2][A-] - k2[HA][O2-]
% dO2- = k3[HO2][A-] - k2[HA][O2-]
% dHO2 = k2[HA][O2-] - k3[HO2][A-] - k1[HO2][Fe(II)]
% dA- = k2[HA][O2-] - k3[HO2][A-]
% dFe(II) = -k1[HO2][Fe(II)]
%dFe(III)OOH = k1[HO2][Fe(II)]

dy = zeros(6,1);
dy(1) = k(3)*y(3)*y(4) - k(2)*y(1)*y(2);
dy(2) = k(3)*y(3)*y(4)- k(2)*y(1)*y(2);
dy(3) = k(2)*y(1)*y(2) - k(3)*y(3)*y(4) - k(1)*y(3)*y(5);
dy(4) = k(2)*y(1)*y(2) - k(3)*y(3)*y(4);
dy(5) = -k(1)*y(3)*y(5);
dy(6) = k(1)*y(3)*y(5);
```