In [5]:
```
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
from sklearn.linear_model import LinearRegression
from sklearn.model_selection import train_test_split, cross_val_score
import warnings
warnings.filterwarnings('ignore')
from sklearn.preprocessing import OneHotEncoder
from sklearn.compose import make_column_transformer
from sklearn.pipeline import make_pipeline
from sklearn.impute import SimpleImputer
from sklearn.preprocessing import Imputer
from sklearn.preprocessing import StandardScaler
from sklearn.linear_model import LinearRegression
from sklearn.linear_model import Ridge
from sklearn.linear_model import Lasso
from sklearn.linear_model import ElasticNet
from sklearn.model_selection import GridSearchCV
```

Task 1 Regression on Ames Housing Dataset

In [6]:
```
df = pd.read_excel('AmesHousing.xls', sheet_name='Sheet1', index_col=0)
df.sample(5)
```

Out[6]:

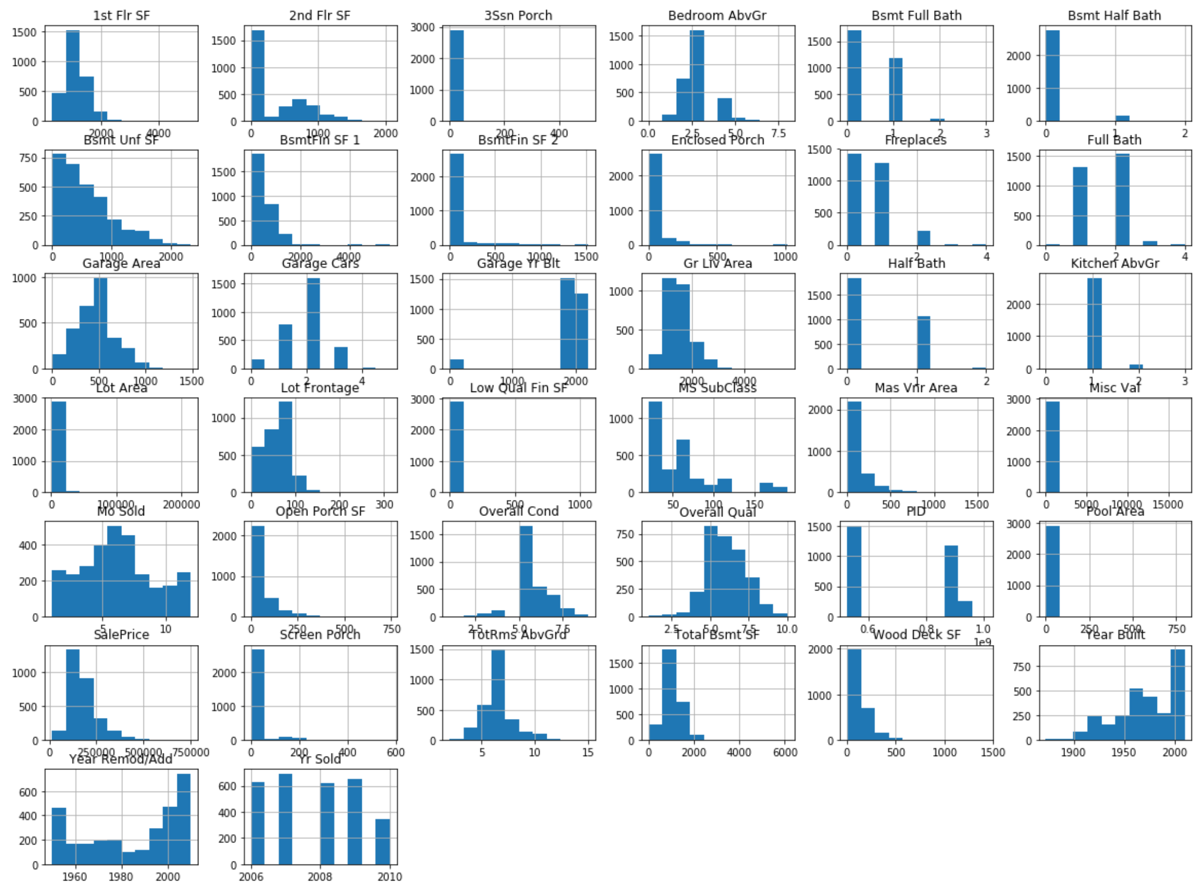
Order	PID	MS SubClass	MS Zoning	Lot Frontage	Lot Area	Street	Alley	Lot Shape	Land Contour	Utilities
1226	534477090	20	RL	80.0	9600	Pave	NaN	Reg	Lvl	AllPub
438	528120020	20	RL	87.0	11146	Pave	NaN	IR1	Lvl	AllPub
2116	906426060	50	RL	NaN	159000	Pave	NaN	IR2	Low	AllPub
1138	531376010	60	RL	65.0	8366	Pave	NaN	IR1	Lvl	AllPub
1190	534129060	20	RL	NaN	15387	Pave	NaN	IR1	Lvl	AllPub

5 rows × 81 columns

1.1

In [7]:
```
dg = df.loc[:, df.dtypes != object]
dg.fillna(0, inplace=True)
dg.hist(figsize=(20,15));
```

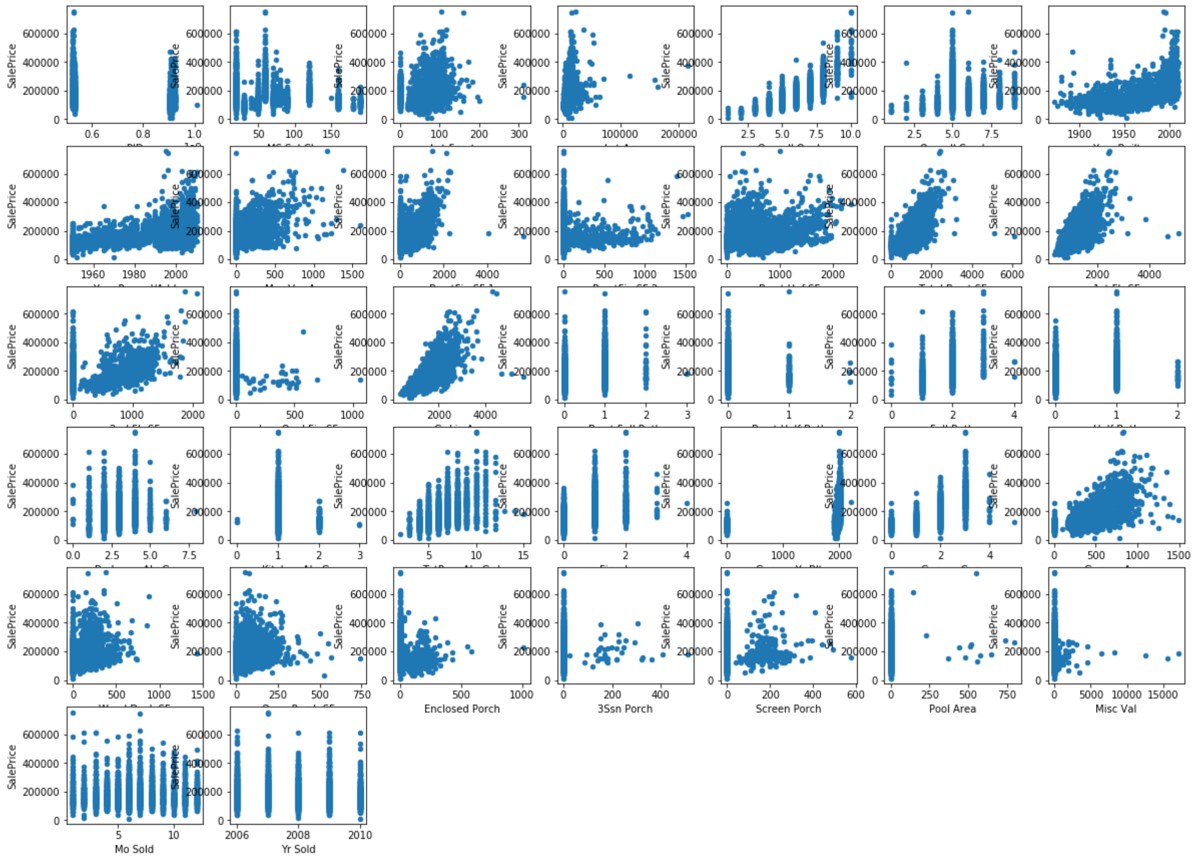
```
C:\Users\Paridhi\Anaconda3\lib\site-packages\pandas\core\frame.py:4034: Setti
ngWithCopyWarning:
A value is trying to be set on a copy of a slice from a DataFrame

See the caveats in the documentation: http://pandas.pydata.org/pandas-docs/st
able/indexing.html#indexing-view-versus-copy
  downcast=downcast, **kwargs)
```



We see there are quite a few columns having extremely high numbers with large values as 0. There are different units and scales and thus due to these two reasons, we need to standardize the data.
There are concentrated distributions, skewed distibutions, discrete variable, linear and non-linear effects seen as well

1.2

In [8]:
```
fig = plt.figure(figsize=(20,15))
for i in range(len(list(dg.columns)[:-1])):
    axi = fig.add_subplot(6, 7, i+1)
    dg.plot.scatter(x = list(dg.columns)[i], y = "SalePrice", ax=axi) ## Cut d
own ylabels
```



1.3

In [9]:
```
X_train, X_test, y_train, y_test = train_test_split(df.iloc[:,df.columns!='Sal
ePrice'], df['SalePrice'])

ds = X_train.loc[:,X_train.dtypes == 'object']
ds.fillna('others', inplace = True)
ds.sample(7)
```

Out[9]:

	MS Zoning	Street	Alley	Lot Shape	Land Contour	Utilities	Lot Config	Land Slope	Neighborhood	Conditic
Order										
1022	RL	Pave	others	Reg	Lvl	AllPub	Inside	Gtl	NWAmes	Nor
1884	RL	Pave	others	Reg	Lvl	AllPub	Inside	Gtl	NAmes	Nor
2525	RL	Pave	others	IR1	Lvl	AllPub	Inside	Gtl	NWAmes	Nor
2178	RL	Pave	others	Reg	Lvl	AllPub	Corner	Gtl	Edwards	Nor
866	RL	Pave	others	Reg	Lvl	AllPub	Inside	Gtl	CollgCr	Nor
620	RL	Pave	others	Reg	Lvl	AllPub	Inside	Gtl	NAmes	Nor
1817	RL	Pave	others	IR1	Lvl	AllPub	CulDSac	Gtl	SawyerW	Nor

7 rows × 43 columns

In [23]:
```
list(ds.columns)[1]
```

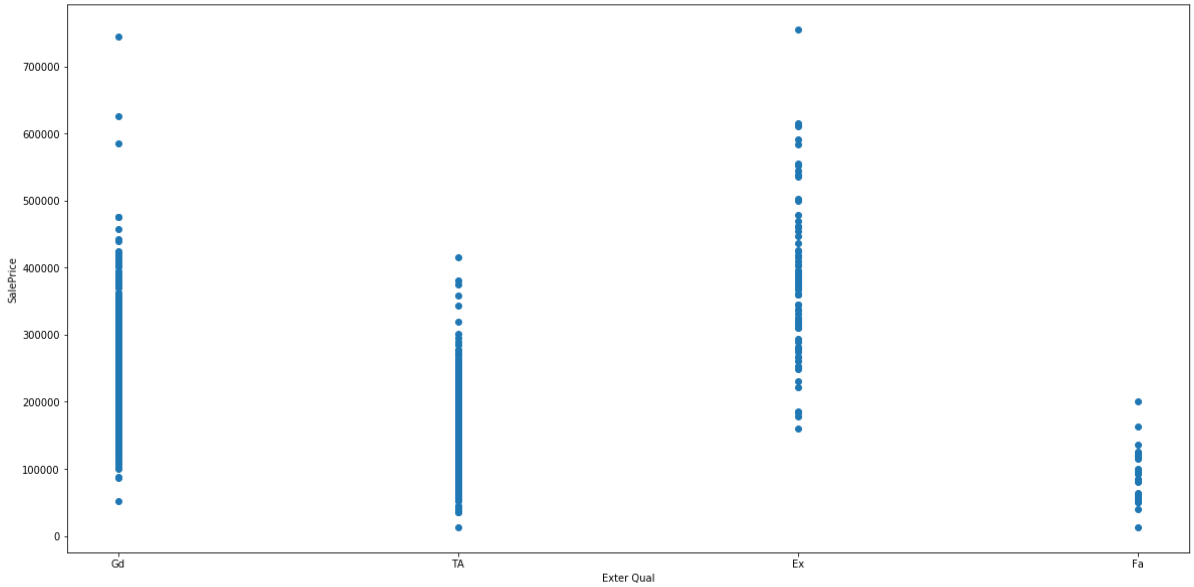
Out[23]: 'Street'

In [25]:
```
from sklearn.preprocessing import OneHotEncoder
counter = index = -1
for i in range(len(list(ds.columns))):
    enc = OneHotEncoder(handle_unknown='ignore')
    temp = np.array(ds.iloc[:,i]).reshape(-1,1)
    enc.fit(temp)
    onehotenc = enc.transform(temp).toarray()
    scores = cross_val_score(LinearRegression(), onehotenc, y_train, cv=10)
    score = np.mean(scores)
    if score > counter:
        counter = score
        index = i

print(counter)
print(list(ds.columns)[index])
```

```
0.49235654778103
Exter Qual
```

In [44]:
```
fig, ax = plt.subplots(1, 1, figsize = (20, 10))
ax.scatter(ds.iloc[:,index], y_train)
ax.set_ylabel('SalePrice')
ax.set_xlabel(list(ds.columns)[index])
```

Out[44]: Text(0.5,0,'Exter Qual')



1.4

Objective is to notice the difference Standardization makes
We do once without Standardization and then we Standardize the data using StandardScalar and compare the results

Without standardScalar

In [ ]:
```
x_data = df.iloc[:,df.columns!='SalePrice']
for i in list(x_data.columns):
    if x_data[i].dtype == 'object':
        x_data[i].fillna('others', inplace = True)
x_data.fillna(x_data.median(), inplace = True)
y_data = df['SalePrice']
X_train, X_test, y_train, y_test = train_test_split(x_data, y_data)

discrete = X_train.dtypes == 'object'
col_trans = make_column_transformer((OneHotEncoder(handle_unknown='ignore'), d
iscrete))

Linear_pipe = make_pipeline(col_trans, LinearRegression())
Linear_fin = np.mean(cross_val_score(Linear_pipe, X_train, y_train, cv=10))

Ridge_pipe = make_pipeline(col_trans, Ridge())
Ridge_fin = np.mean(cross_val_score(Ridge_pipe, X_train, y_train, cv=10))

Lasso_pipe = make_pipeline(col_trans, Lasso())
Lasso_fin = np.mean(cross_val_score(Lasso_pipe, X_train, y_train, cv=10))

ElasticNet_pipe = make_pipeline(col_trans, ElasticNet())
ElasticNet_fin = np.mean(cross_val_score(ElasticNet_pipe, X_train, y_train, cv
=10))
```

In [48]:
```
print('Linear: {:.2f}' .format(Linear_fin))
print('Ridge: {:.2f}'.format(Ridge_fin))
print('Lasso: {:.2f}'.format(Lasso_fin))
print('ElasticNet: {:.2f}'.format(ElasticNet_fin))
```

```
Linear: 0.79
Ridge: 0.80
Lasso: 0.80
ElasticNet: 0.64
```

With StandardScalar

In [ ]:
```
col_trans = make_column_transformer((StandardScaler(), ~discrete), (OneHotEnco
der(handle_unknown='ignore'), discrete))

Linear_pipe = make_pipeline(col_trans, LinearRegression())
Linear_fin = np.mean(cross_val_score(Linear_pipe, X_train, y_train, cv=10))

Ridge_pipe = make_pipeline(col_trans, Ridge())
Ridge_fin = np.mean(cross_val_score(Ridge_pipe, X_train, y_train, cv=10))

Lasso_pipe = make_pipeline(col_trans, Lasso())
Lasso_fin = np.mean(cross_val_score(Lasso_pipe, X_train, y_train, cv=10))

ElasticNet_pipe = make_pipeline(col_trans, ElasticNet())
ElasticNet_fin = np.mean(cross_val_score(ElasticNet_pipe, X_train, y_train, cv
=10))
```

In [51]:
```
print('Linear: {:.2f}' .format(Linear_fin))
print('Ridge: {:.2f}'.format(Ridge_fin))
print('Lasso: {:.2f}'.format(Lasso_fin))
print('ElasticNet: {:.2f}'.format(ElasticNet_fin))
```

```
Linear: 0.83
Ridge: 0.83
Lasso: 0.84
ElasticNet: 0.81
```

Noticably, the scores have improveda lot!

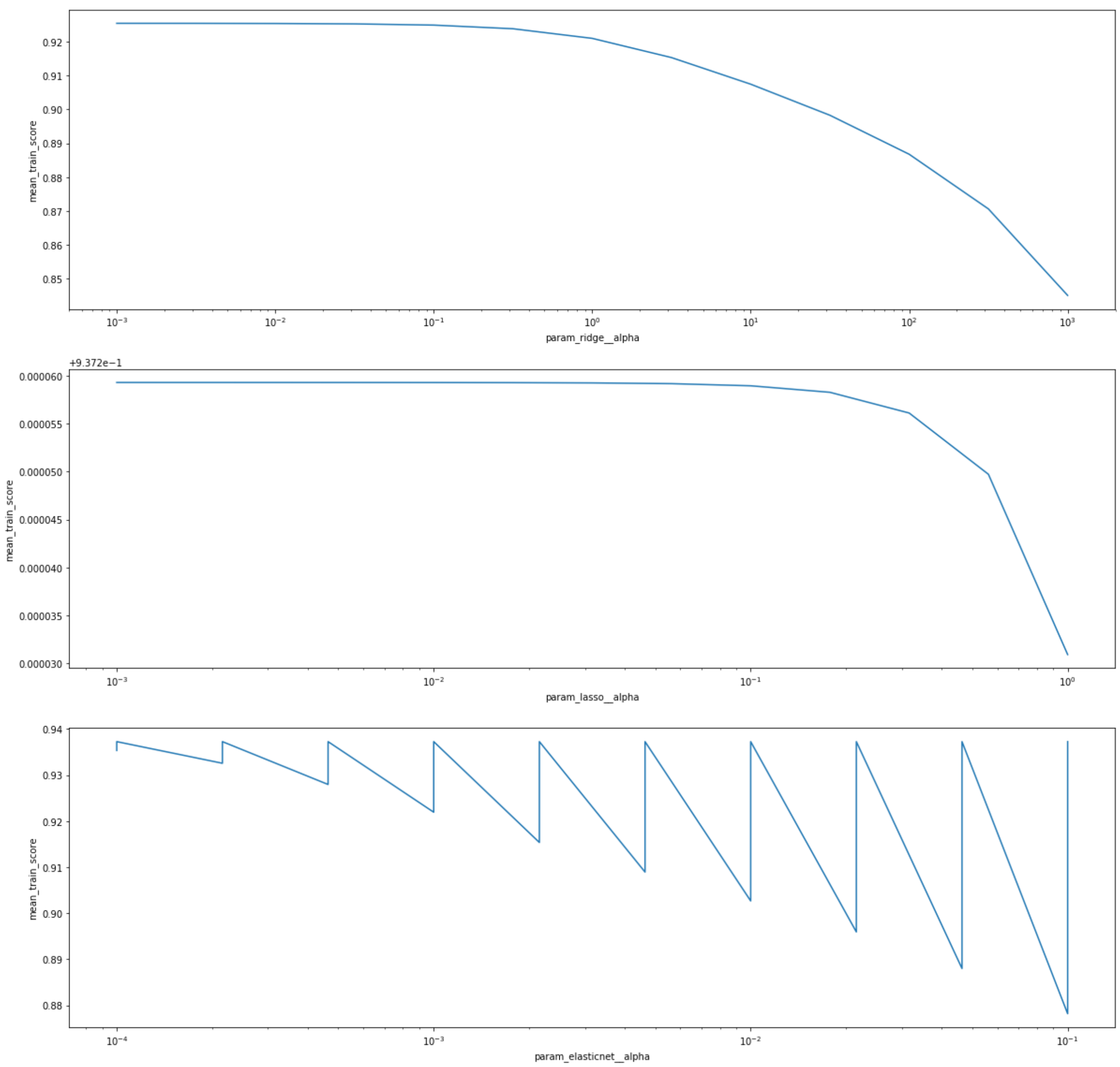
1.5

In [ ]:
```
Ridgy = {'ridge__alpha': np.logspace(-3, 3, 13)}
Lassy = {'lasso__alpha': np.logspace(-3, 0, 13)}
ElasticNety = {'elasticnet__alpha': np.logspace(-4, -1, 10),
               'elasticnet__l1_ratio': [0.01, .1, .5, .9, 1]}

Ridge_got = GridSearchCV(Ridge_pipe, Ridgy, cv=10)
Lasso_got = GridSearchCV(Lasso_pipe, Lassy, cv=10)
ElasticNet_got = GridSearchCV(ElasticNet_pipe, ElasticNety, cv=10)

Ridge_got.fit(X_train, y_train)
Lasso_got.fit(X_train, y_train)
ElasticNet_got.fit(X_train, y_train)
```

In [53]:
```
print('Ridge: {:.2f}'.format(Ridge_got.best_score_))
print('Lasso: {:.2f}'.format(Lasso_got.best_score_))
print('ElasticNet: {:.2f}'.format(ElasticNet_got.best_score_))
```

```
Ridge: 0.83
Lasso: 0.84
ElasticNet: 0.84
```

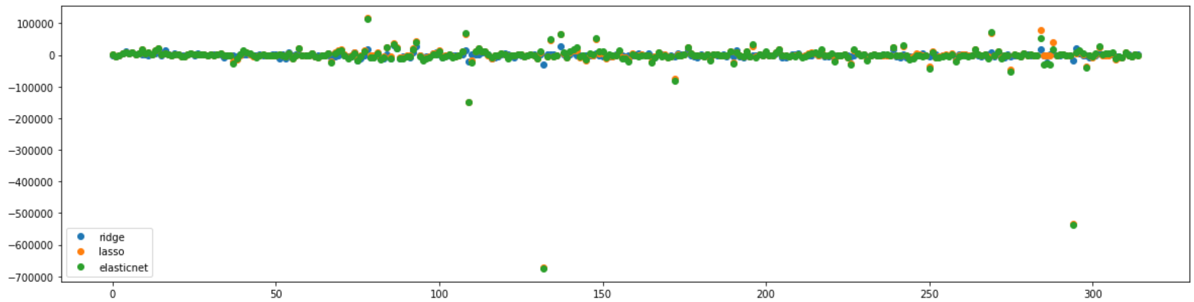
In [59]:
```
fig, ax = plt.subplots(3,1, figsize = (20, 20))
ax[0].plot(list(Ridge_got.cv_results_['param_ridge__alpha']),
        list(Ridge_got.cv_results_['mean_train_score']))
ax[0].set_ylabel('mean_train_score')
ax[0].set_xlabel('param_ridge__alpha')
ax[0].set_xscale("log")

ax[1].plot(list(Lasso_got.cv_results_['param_lasso__alpha']),
      list(Lasso_got.cv_results_['mean_train_score']))
ax[1].set_ylabel('mean_train_score')
ax[1].set_xlabel('param_lasso__alpha')
ax[1].set_xscale("log")

ax[2].plot(list(ElasticNet_got.cv_results_['param_elasticnet__alpha']),
      list(ElasticNet_got.cv_results_['mean_train_score']))
ax[2].set_ylabel('mean_train_score')
ax[2].set_xlabel('param_elasticnet__alpha')
ax[2].set_xscale("log")
```



1.6

In [67]:
```
plt.figure(figsize=(20, 5))
plt.plot(Ridge_got.best_estimator_.named_steps['ridge'].coef_, 'o', label='ridge')
plt.plot(Lasso_got.best_estimator_.named_steps['lasso'].coef_, 'o', label='lasso')
plt.plot(ElasticNet_got.best_estimator_.named_steps['elasticnet'].coef_, 'o', label='elasticnet')
plt.legend()
```

Out[67]: <matplotlib.legend.Legend at 0x25a37652f28>



Yes they seem to be agreeing on which features are important as can be seen by the simultaneous rise and fall of the coefficients in all three.