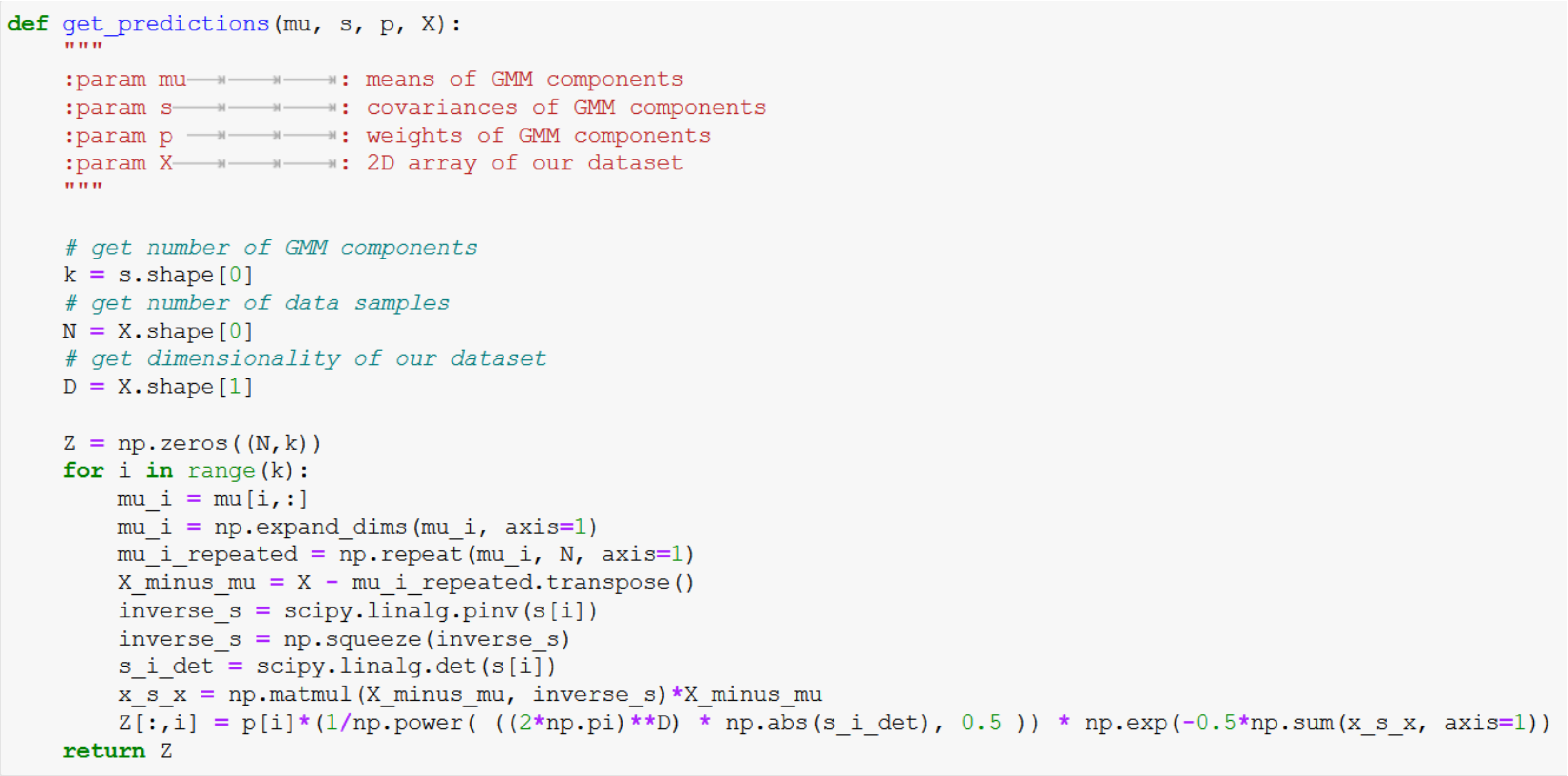
Programming equations

$$p(x|\mu,\Sigma) = \frac{1}{(2\pi)^{d/2}|\Sigma|^{1/2}} e^{-\frac{1}{2}(x-\mu)^T\Sigma^{-1}(x-\mu)}$$

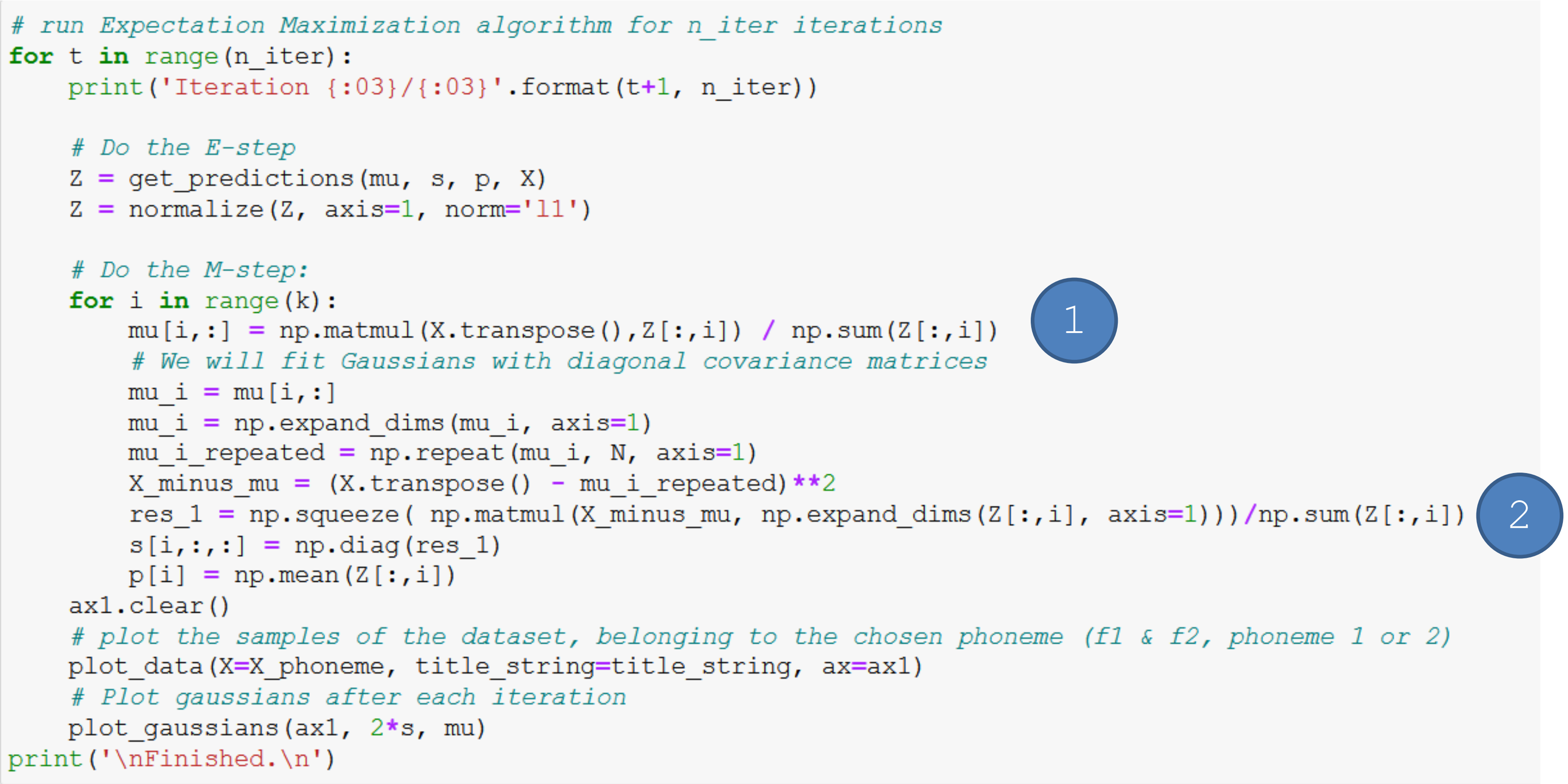


```python
def get_predictions(mu, s, p, X):
    """
    :param mu           : means of GMM components
    :param s            : covariances of GMM components
    :param p            : weights of GMM components
    :param X            : 2D array of our dataset
    """

    # get number of GMM components
    k = s.shape[0]
    # get number of data samples
    N = X.shape[0]
    # get dimensionality of our dataset
    D = X.shape[1]

    Z = np.zeros((N,k))
    for i in range(k):
        mu_i = mu[i,:]
        mu_i = np.expand_dims(mu_i, axis=1)
        mu_i_repeated = np.repeat(mu_i, N, axis=1)
        X_minus_mu = X - mu_i_repeated.transpose()
        inverse_s = scipy.linalg.pinv(s[i])
        inverse_s = np.squeeze(inverse_s)
        s_i_det = scipy.linalg.det(s[i])
        x_s_x = np.matmul(X_minus_mu, inverse_s)*X_minus_mu
        Z[:,i] = p[i]*(1/np.power( ((2*np.pi)**D) * np.abs(s_i_det), 0.5 )) * np.exp(-0.5*np.sum(x_s_x, axis=1))
    return Z
```

Programming equations

$$\mu_i = \frac{1}{\mu_i}\sum_i x_i Z_i$$ 1

$$p_i = \frac{1}{\mu_i}\sum Z_i (X-\mu_i)^T (X-\mu_i)$$ 2



```
# run Expectation Maximization algorithm for n_iter iterations
for t in range(n_iter):
    print('Iteration {:03}/{:03}'.format(t+1, n_iter))

    # Do the E-step
    Z = get_predictions(mu, s, p, X)
    Z = normalize(Z, axis=1, norm='l1')

    # Do the M-step:
    for i in range(k):
        mu[i,:] = np.matmul(X.transpose(),Z[:,i]) / np.sum(Z[:,i])
        # We will fit Gaussians with diagonal covariance matrices
        mu_i = mu[i,:]
        mu_i = np.expand_dims(mu_i, axis=1)
        mu_i_repeated = np.repeat(mu_i, N, axis=1)
        X_minus_mu = (X.transpose() - mu_i_repeated)**2
        res_1 = np.squeeze( np.matmul(X_minus_mu, np.expand_dims(Z[:,i], axis=1)))/np.sum(Z[:,i])
        s[i,:,:] = np.diag(res_1)
        p[i] = np.mean(Z[:,i])
    ax1.clear()
    # plot the samples of the dataset, belonging to the chosen phoneme (f1 & f2, phoneme 1 or 2)
    plot_data(X=X_phoneme, title_string=title_string, ax=ax1)
    # Plot gaussians after each iteration
    plot_gaussians(ax1, 2*s, mu)
print('\nFinished.\n')
```

1

2