ExponentialSum (Partial)

```
def exponentialSum(self):
    S = [a for a in range(self.k) if self.D[a] in [2, 6]]
    if len(S) != 0:
        a = S[0]

        # Construct R as in comment on page 12
        R = np.identity(self.k)
        for b in S[1:]:
            R[b, a] += 1
        R = R % 2

        self.updateDJ(R)
        S = [a]
    # Now J[a, a] = 0 for all a not in S

    E = [k for k in range(self.k) if k not in S]
    M = []
    Dimers = []  # maintain list of dimers rather than r

    while len(E) > 0:
        a = E[0]
        K = [b for b in E[1:] if self.J[a, b] == 4]

        if len(K) == 0:  # found a new monomer {a}
            M.append(a)
            E = E[1:]
        else:
            b = K[0]

            # Construct R for basis change
            R = np.identity(self.k)
            for c in [x for x in E if x != a and x != b]:
                if self.J[a, c] == 4: R[c, a] += 1
                if self.J[b, c] == 4: R[c, b] += 1
            R = R % 2

            self.updateDJ(R)

            # {a, b} form a new dimer
            Dimers.append([a, b])
            E = [x for x in E if x != a and x != b]

    if len(S) != 0:
        # Compute W(K,q) from Eq. 63
        raise NotImplementedError  # Where exactly in reference 15?
    else:
        # Compute W_0, W_1 from Eq. 68
        raise NotImplementedError

# evaluates the expression in the comment on page 12
def W(p, m, eps):
    return eps * 2**(p/2) * np.exp(1j*np.pi*m/4)
```