Thompson's Problem

October 1, 2022

1 Import the packages needed

[1]:
```
import numpy as np
import matplotlib.pyplot as plt
```

2 The electrostatic potential energy

As for the electrostatic potential energy, we have:

$$U_E(N) = \frac{1}{2}\sum_{i=1}^{N}\sum_{j\neq i}^{N}\frac{1}{||x_i - x_j||_2}$$

[2]:
```
def U_E(x_N):

    # x_N means the N positions of the particles

    U_E = 0

    for i in range(len(x_N)):
        U_E += (1/2) * (1 / np.linalg.norm(x_N[i, :] - np.delete(x_N, i, 0),␣
↪axis = 1)).sum()

    return U_E
```

3 The gradient of the electrostatic potential energy

As for the gradient, we first have:

$$\begin{aligned}\frac{\partial(\frac{1}{||x_i - x_j||_2})}{\partial x_i} &= \frac{\partial(\frac{1}{||x_i - x_j||_2})}{\partial(||x_i - x_j||_2)} \cdot \frac{\partial||x_i - x_j||_2}{\partial x_i} \\ &= -(||x_i - x_j||_2)^{-2} \cdot \frac{x_i - x_j}{||x_i - x_j||_2} \\ &= -\frac{x_i - x_j}{||x_i - x_j||_2^3}\end{aligned}$$

Then we have:

$$\frac{\partial U_E(N)}{\partial x_i} = \sum_{j \neq i}^{N} (-\frac{1}{2}) \frac{x_i - x_j}{||x_i - x_j||_2^3}.$$

[3]:
```
def grad_of_U(x_N):

    grad = np.zeros_like(x_N)

    for i in range(len(x_N)):
        grad_x_i = np.zeros_like(x_N[i, :])

        for k in range(len(x_N)):
            if k != i:
                grad_x_i += (-1/2) * (x_N[i, :] - x_N[k, :]) / (np.linalg.
↪norm(x_N[i, :] - x_N[k, :]) ** 3)

        grad[i, :] = grad_x_i

    return grad
```

4 Projected Gradient Descent (PGD)

[4]:
```
def PGD(X, lr, beta, accuracy):

    while True:
        # Initialize the learning rate (or step size).
        lr_ = lr

        while True:
            # Do gradient descent.
            X_next = X - lr_ * grad_of_U(X)

            # Do projection.
            X_next = X_next / np.linalg.norm(X_next, axis=1).reshape(-1, 1)

            # Calculate the energy.
            U_next = U_E(X_next)
            U_now = U_E(X)

            # Do backtracking line search.
            if U_next < U_now :
                break
            else:
                lr_ *= beta

        # End the iteration based on accuracy.
```

```
        if abs(U_next - U_now) < accuracy:
            break

        X = X_next

    return X
```

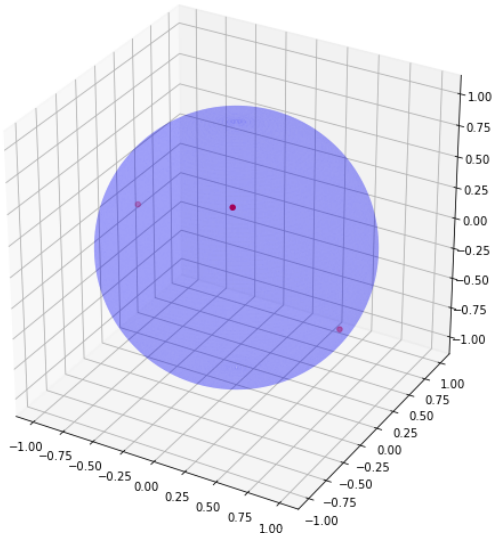
5 N from 3 to 5

[5]:
```
fig, axs = plt.subplots(3, 1, figsize=(30, 30),␣
↪subplot_kw=dict(projection='3d'))

for n, ax in zip(range(3, 16), axs.ravel()):
    X = np.random.randn(n, 3)
    X /= np.linalg.norm(X, axis=1).reshape(-1, 1)
    X_final = PGD(X, 0.1, 0.7, 1e-8)

    print('The coordinates when N = {N}: '.format(N=n))
    for i in range(n):
        print(X_final[i])
    print('\n')

    potential = U_E(X_final)

    phi, theta = np.mgrid[0.0:2 * np.pi:200j, 0.0:2.0 * np.pi: 200j]
    X = np.sin(phi) * np.cos(theta)
    Y = np.sin(phi) * np.sin(theta)
    Z = np.cos(phi)

    ax.plot_surface(X, Y, Z, rstride=1, cstride=1, alpha=0.1, color='blue',␣
↪shade=0)
    ax.scatter(X_final[:, 0], X_final[:, 1], X_final[:, 2], c='r')
    ax.set_box_aspect([1, 1, 1])
    ax.set_title(f"The electrostatic Potential Energy when $N={n}$: U =␣
↪{potential:.6f}")

plt.show()
```

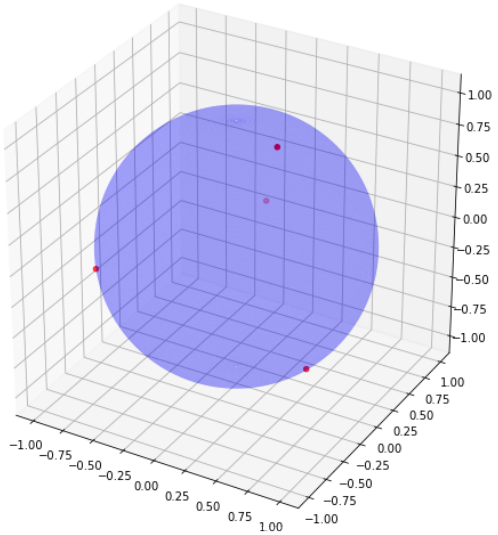
```
The coordinates when N = 3:
[ 0.64009847  0.36773651 -0.67456935]
[ 0.32728372 -0.61762005  0.71514393]
[-0.96768469  0.2487736  -0.0412073 ]


The coordinates when N = 4:
[-0.29087949  0.94215425 -0.16653677]
```

```
[ 0.36591175 -0.07753022  0.9274145 ]
[ 0.70835431 -0.25137835 -0.65957797]
[-0.78338504 -0.61322964 -0.10127822]


The coordinates when N = 5:
[-0.6511418   0.74654091 -0.13671515]
[-0.71074913 -0.66359873 -0.23339325]
[0.71186044 0.66260047 0.23284187]
[ 0.55478165 -0.32600768 -0.76546477]
[ 0.09511056 -0.41941091  0.90280035]
```

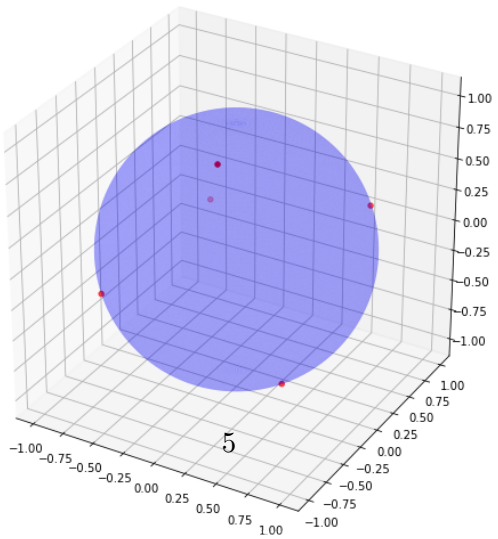
The electrostatic Potential Energy when $N = 3$: U = 1.732051



The electrostatic Potential Energy when $N = 4$: U = 3.674235



The electrostatic Potential Energy when $N = 5$: U = 6.474692



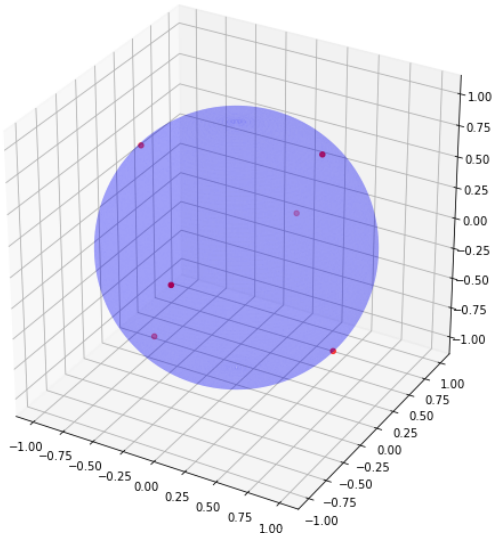
6 N from 6 to 8

[6]:
```
fig, axs = plt.subplots(3, 1, figsize=(30, 30),␣
 ↪subplot_kw=dict(projection='3d'))

for n, ax in zip(range(6, 9), axs.ravel()):
    X = np.random.randn(n, 3)
    X /= np.linalg.norm(X, axis=1).reshape(-1, 1)
    X_final = PGD(X, 0.1, 0.7, 1e-8)

    print('The coordinates when N = {N}: '.format(N=n))
    for i in range(n):
        print(X_final[i])
    print('\n')

    potential = U_E(X_final)

    phi, theta = np.mgrid[0.0:2 * np.pi:200j, 0.0:2.0 * np.pi: 200j]
    X = np.sin(phi) * np.cos(theta)
    Y = np.sin(phi) * np.sin(theta)
    Z = np.cos(phi)

    ax.plot_surface(X, Y, Z, rstride=1, cstride=1, alpha=0.1, color='blue',␣
 ↪shade=0)
    ax.scatter(X_final[:, 0], X_final[:, 1], X_final[:, 2], c='r')
    ax.set_box_aspect([1, 1, 1])
    ax.set_title(f"The electrostatic Potential Energy when $N={n}$: U =␣
 ↪{potential:.6f}")

plt.show()
```

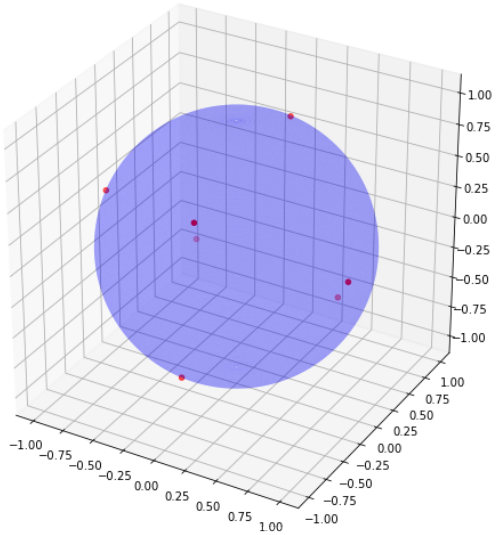
```
The coordinates when N = 6:
[-0.05448286  0.97568733 -0.21228718]
[ 0.05446286 -0.97585119  0.21153781]
[-0.56404828 -0.20522748 -0.79983199]
[-0.82387979  0.07601725  0.56164355]
[0.56370876 0.20519072 0.80008075]
[ 0.82423931 -0.07581663 -0.56114294]


The coordinates when N = 7:
[-0.68197257  0.59226097 -0.42911579]
[ 0.12303654 -0.78213466  0.61084153]
[0.07148688 0.64205585 0.7633177 ]
```

```
[ 0.98122137 -0.17127591  0.08870839]
[-0.90130434 -0.27474687  0.3349099 ]
[-0.07277024 -0.64132697 -0.76380902]
[ 0.48019096  0.63523085 -0.60489537]


The coordinates when N = 8:
[-0.48590164  0.21454197 -0.84727289]
[ 0.79193896 -0.48944262  0.36507344]
[-0.91342125  0.3306815   0.23730014]
[ 0.15788725  0.98654191 -0.04250506]
[-0.46621772 -0.7128414   0.52392574]
[ 0.77868162  0.16808158 -0.60448616]
[0.1734384  0.3302182  0.92783353]
[-0.03640563 -0.82778114 -0.55986874]
```

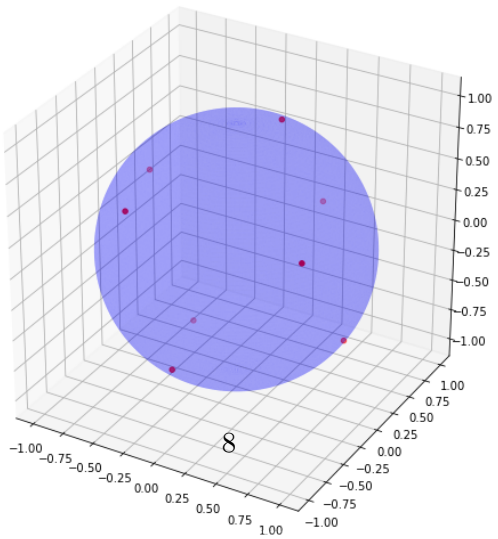
The electrostatic Potential Energy when $N = 6$: U = 9.985282



The electrostatic Potential Energy when $N = 7$: U = 14.452988



The electrostatic Potential Energy when $N = 8$: U = 19.675288



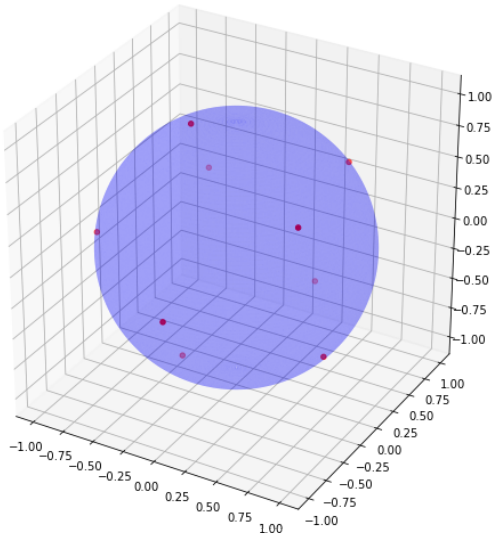
7 N from 9 to 11

[7]:
```
fig, axs = plt.subplots(3, 1, figsize=(30, 30),␣
 ↪subplot_kw=dict(projection='3d'))

for n, ax in zip(range(9, 12), axs.ravel()):
    X = np.random.randn(n, 3)
    X /= np.linalg.norm(X, axis=1).reshape(-1, 1)
    X_final = PGD(X, 0.1, 0.7, 1e-8)

    print('The coordinates when N = {N}: '.format(N=n))
    for i in range(n):
        print(X_final[i])
    print('\n')

    potential = U_E(X_final)

    phi, theta = np.mgrid[0.0:2 * np.pi:200j, 0.0:2.0 * np.pi: 200j]
    X = np.sin(phi) * np.cos(theta)
    Y = np.sin(phi) * np.sin(theta)
    Z = np.cos(phi)

    ax.plot_surface(X, Y, Z, rstride=1, cstride=1, alpha=0.1, color='blue',␣
 ↪shade=0)
    ax.scatter(X_final[:, 0], X_final[:, 1], X_final[:, 2], c='r')
    ax.set_box_aspect([1, 1, 1])
    ax.set_title(f"The electrostatic Potential Energy when $N={n}$: U =␣
 ↪{potential:.6f}")

plt.show()
```

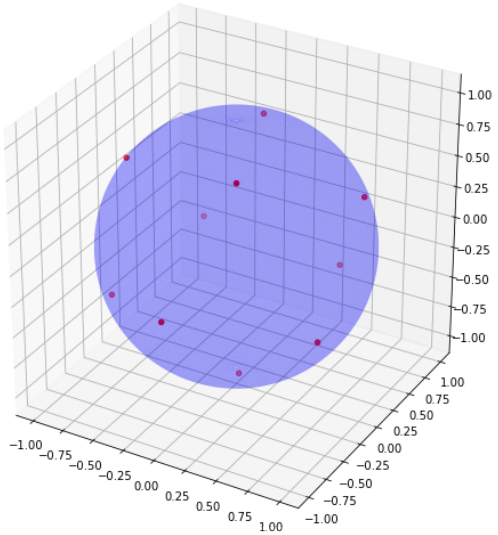
```
The coordinates when N = 9:
[ 0.71605901 -0.40824715  0.56621   ]
[ 0.81473859 -0.19807712 -0.54494632]
[ 0.21957172  0.76681424 -0.60314524]
[0.50941143 0.70551651 0.49269306]
[-0.933164   -0.3575635   0.03678714]
[-0.30732453 -0.09454146  0.9468968 ]
[-0.66066344  0.74187924  0.11462559]
[-0.00269088 -0.99608269 -0.08838571]
[-0.35600796 -0.15972659 -0.92073109]


The coordinates when N = 10:
```

```
[-0.67322283  0.68517339 -0.27804397]
[-0.75496367 -0.22906607  0.61445797]
[ 0.86596437 -0.38559343 -0.31847042]
[0.81584145 0.36301608 0.45013559]
[ 0.28531794 -0.49387767  0.82138816]
[-0.81448402 -0.3668367  -0.44949596]
[ 0.06464475 -0.07648841 -0.99497265]
[-0.01540026 -0.99463438 -0.10229993]
[ 0.4025365   0.80305694 -0.43939039]
[-0.17628671  0.69539793  0.69666687]


The coordinates when N = 11:
[ 0.64627534 -0.69824982  0.30785608]
[0.63921198 0.27076736 0.71978683]
[-0.37907182  0.53825296  0.7527206 ]
[ 0.90255096  0.21997724 -0.37015102]
[-0.3823105  -0.89932201 -0.21227011]
[-0.24226568  0.14547094 -0.95924217]
[-0.9840523  -0.1723772   0.04389964]
[ 0.40026813 -0.56201072 -0.72382966]
[-0.25013287 -0.50337508  0.82707139]
[-0.65024328  0.71213186 -0.26467317]
[ 0.30299018  0.94679761 -0.10849528]
```

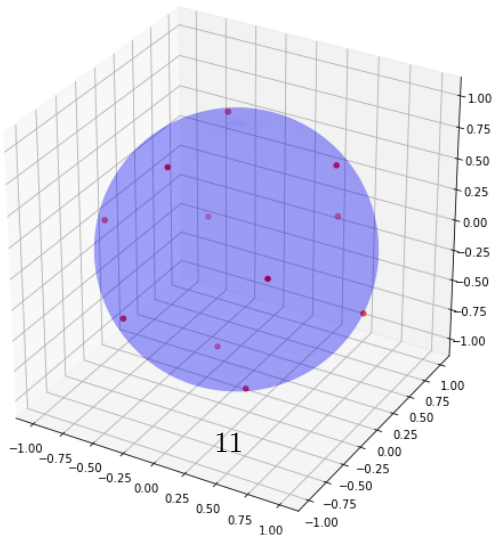
The electrostatic Potential Energy when $N = 9$: U = 25.759987



The electrostatic Potential Energy when $N = 10$: U = 32.716951



The electrostatic Potential Energy when $N = 11$: U = 40.596452



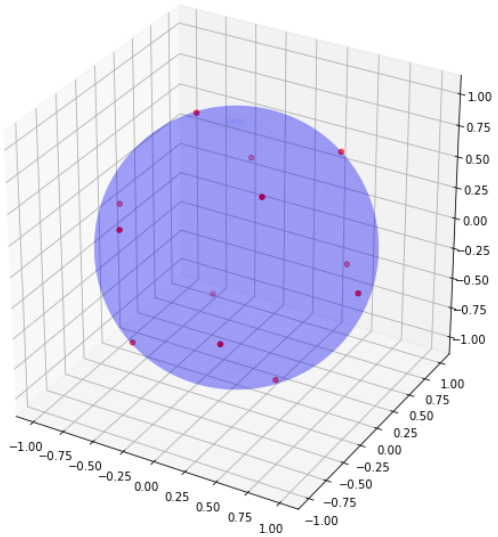
8 N from 12 to 14

[8]:
```
fig, axs = plt.subplots(3, 1, figsize=(30, 30),␣
 ↪subplot_kw=dict(projection='3d'))

for n, ax in zip(range(12, 15), axs.ravel()):
    X = np.random.randn(n, 3)
    X /= np.linalg.norm(X, axis=1).reshape(-1, 1)
    X_final = PGD(X, 0.1, 0.7, 1e-8)

    print('The coordinates when N = {N}: '.format(N=n))
    for i in range(n):
        print(X_final[i])
    print('\n')

    potential = U_E(X_final)

    phi, theta = np.mgrid[0.0:2 * np.pi:200j, 0.0:2.0 * np.pi: 200j]
    X = np.sin(phi) * np.cos(theta)
    Y = np.sin(phi) * np.sin(theta)
    Z = np.cos(phi)

    ax.plot_surface(X, Y, Z, rstride=1, cstride=1, alpha=0.1, color='blue',␣
 ↪shade=0)
    ax.scatter(X_final[:, 0], X_final[:, 1], X_final[:, 2], c='r')
    ax.set_box_aspect([1, 1, 1])
    ax.set_title(f"The electrostatic Potential Energy when $N={n}$: U =␣
 ↪{potential:.6f}")

plt.show()
```

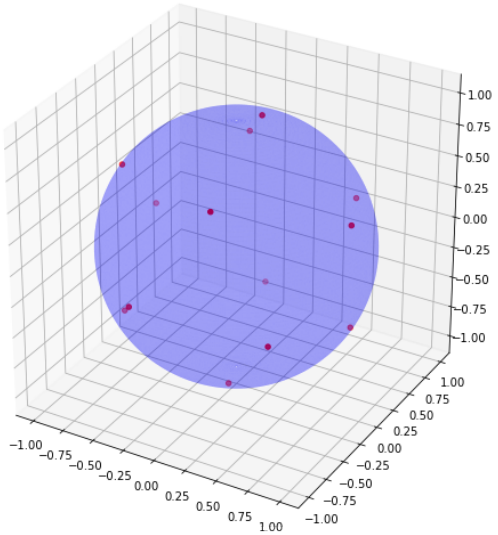
```
The coordinates when N = 12:
[-0.46768889 -0.78874613  0.39893189]
[-0.39518641  0.12727876  0.90974052]
[-0.39308059  0.89853205  0.19526343]
[-0.9974263   0.05028301  0.05111157]
[ 0.46426019 -0.45922948  0.75734455]
[0.51036469 0.58355243 0.63166008]
[ 0.46751058  0.78900697 -0.39862497]
[-0.46414772  0.45913302 -0.75747196]
[ 0.39303823 -0.89857839 -0.19513541]
[ 0.99740872 -0.05021794 -0.05151709]
[-0.51018097 -0.58386603 -0.63151868]
[ 0.39512847 -0.12714826 -0.90978393]
```

```
The coordinates when N = 13:
[ 0.09950879 -0.28384197 -0.95369373]
[-0.35892055  0.81493048  0.45504324]
[0.09020072 0.1998243  0.97567109]
[ 0.20509548 -0.70371313  0.68023796]
[-0.30441556 -0.94800094 -0.09287292]
[-0.15843396  0.70118039 -0.69515807]
[0.49374722 0.86230494 0.11244497]
[ 0.78238108  0.25815745 -0.5667756 ]
[-0.90911773  0.40473395 -0.09846515]
[-0.74574283 -0.30847871 -0.59051546]
[0.9051555  0.00512427 0.42504971]
[-0.75015781 -0.29360345  0.59250339]
[ 0.65865753 -0.7122982  -0.24249027]


The coordinates when N = 14:
[-0.05611052 -0.19944495 -0.97830124]
[-0.20941967  0.5394146   0.81558279]
[-0.90257794 -0.05370736 -0.42716342]
[ 0.21269624 -0.23319577  0.94888357]
[ 0.91053356 -0.3540842   0.21343152]
[ 0.07787042 -0.91593038  0.39371023]
[-0.22065187  0.64098733 -0.73515168]
[ 0.75704195  0.31483327 -0.57250982]
[-0.76650175  0.62954724  0.12706434]
[ 0.17970776  0.9835669  -0.01735705]
[-0.48046203 -0.82631773 -0.29386264]
[-0.75704195 -0.31483327  0.57250982]
[0.74474184 0.48582809 0.45752667]
[ 0.51017393 -0.69666377 -0.50436311]
```

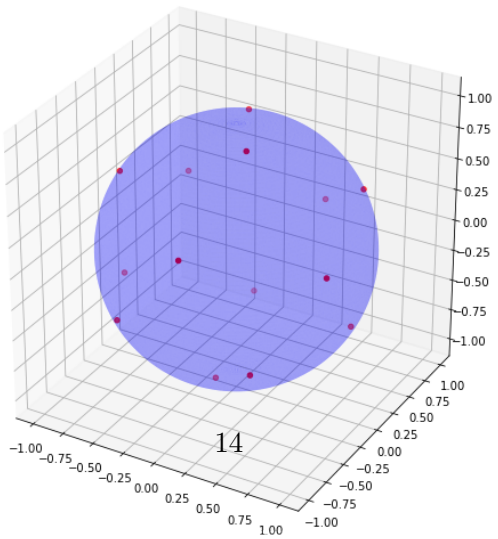
The electrostatic Potential Energy when $N = 12$: U = 49.165253



The electrostatic Potential Energy when $N = 13$: U = 58.853234



The electrostatic Potential Energy when $N = 14$: U = 69.306364

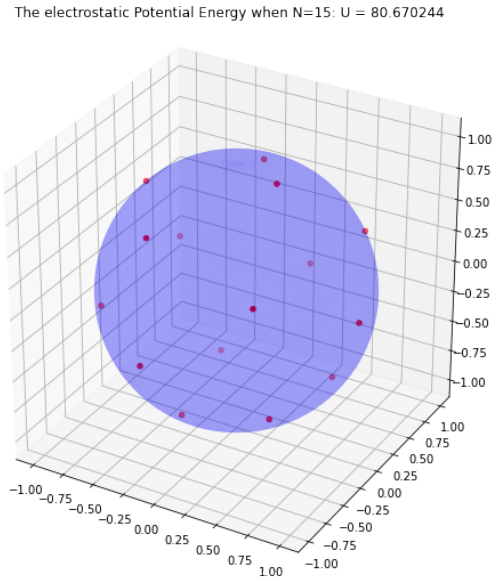


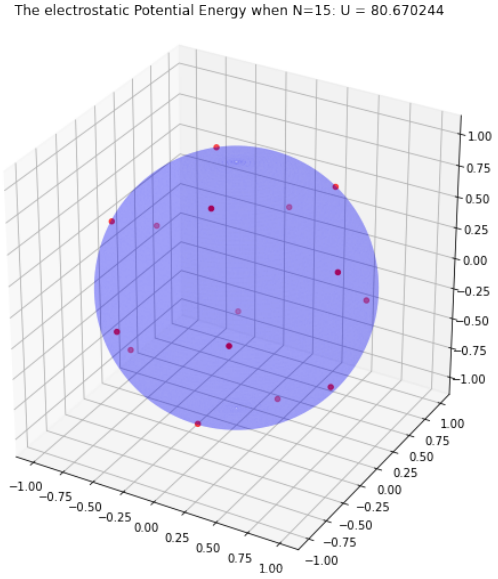
9 N = 15

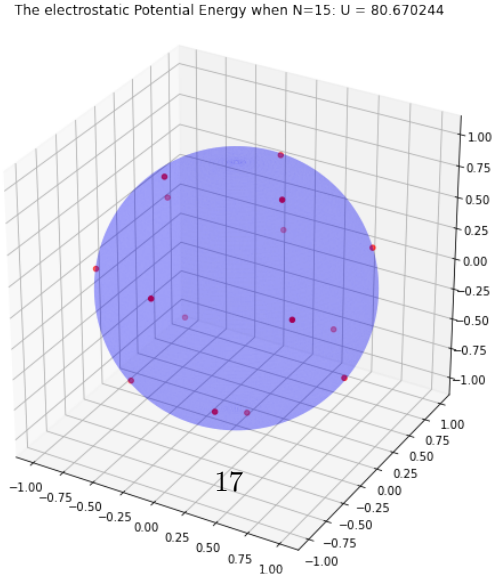
[9]:
```
fig, axs = plt.subplots(3, 1, figsize=(30, 30),␣
 ↪subplot_kw=dict(projection='3d'))

for i in range(len(axs)):
    ax = axs.ravel()[i]
    np.random.seed(i)
    X = np.random.randn(15, 3)
    X /= np.linalg.norm(X, axis=1).reshape(-1, 1)
    X_final = PGD(X, 0.1, 0.7, 1e-9)

    print('The coordinates when N = 15: ')
    for i in range(n):
        print(X_final[i])
    print('\n')

    potential = U_E(X_final)

    phi, theta = np.mgrid[0.0:2 * np.pi:200j, 0.0:2.0 * np.pi: 200j]
    X = np.sin(phi) * np.cos(theta)
    Y = np.sin(phi) * np.sin(theta)
    Z = np.cos(phi)

    ax.plot_surface(X, Y, Z, rstride=1, cstride=1, alpha=0.1, color='blue',␣
 ↪shade=0)
    ax.scatter(X_final[:, 0], X_final[:, 1], X_final[:, 2], c='r')
    ax.set_box_aspect([1, 1, 1])
    ax.set_title(f"The electrostatic Potential Energy when N=15: U = {potential:
 ↪.6f}")

plt.show()
```

```
The coordinates when N = 15:
[ 0.06392245  0.96909031 -0.23828113]
[ 0.58987207 -0.56952662 -0.57244246]
[ 0.9978186  -0.04136053  0.05145232]
[ 0.32798882 -0.01624425  0.94454193]
[-0.3150216  -0.68106893  0.66098903]
[-0.18806531  0.71992339  0.66808813]
[-0.2225157  -0.39327883 -0.89208661]
[ 0.5857473   0.35824652 -0.727021  ]
[ 0.55427485 -0.73770547  0.38544783]
[-0.75023796  0.02919312  0.6605231 ]
```

```
[-0.37447786  0.42299758 -0.82512992]
[-0.91410525 -0.34214144 -0.21760245]
[0.66431422 0.66347223 0.34422553]
[-0.20783236 -0.96199226 -0.17713442]


The coordinates when N = 15:
[ 0.87455927 -0.20738181 -0.43833648]
[ 0.23315505  0.18947987 -0.95380087]
[ 0.88081157 -0.1468006   0.45013395]
[-0.37203634  0.66977581 -0.64264246]
[ 0.07638933 -0.47252765  0.87799903]
[-0.90819195  0.41532337  0.05190257]
[ 0.67422436  0.69305869 -0.25512968]
[0.48404977 0.5477373  0.68240726]
[ 0.45277212 -0.88345738  0.12041786]
[-0.7995383  -0.35588176  0.48382505]
[ 0.07088802 -0.66134941 -0.74672073]
[-0.42643371 -0.9042992  -0.01993114]
[-0.3676026   0.35632912  0.85900983]
[-0.75058907 -0.22049613 -0.62289446]


The coordinates when N = 15:
[ 0.10047189 -0.02081923 -0.99472205]
[ 0.89764819 -0.05172631 -0.43766667]
[ 0.30409534 -0.82206319 -0.48139188]
[-0.01253432  0.64831279  0.76127092]
[-0.12996485 -0.92923045  0.34589002]
[-0.44943817 -0.21453765  0.86716719]
[-0.66332028  0.3945524  -0.63587311]
[ 0.40678028  0.70553802 -0.58029812]
[-0.53572033 -0.56433655 -0.62811463]
[-0.90730878 -0.40821577  0.10075048]
[-0.17030413  0.98421391 -0.04816094]
[0.79354374 0.53494786 0.29003297]
[ 0.7691164  -0.58064263  0.26704699]
[ 0.43752079 -0.1364688   0.88879234]
```


The electrostatic Potential Energy when N=15: U = 80.670244


The electrostatic Potential Energy when N=15: U = 80.670244


The electrostatic Potential Energy when N=15: U = 80.670244

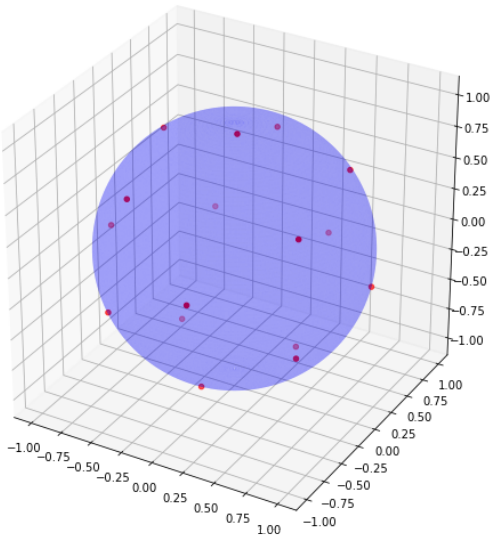
10 N = 16

[10]:
```
fig, axs = plt.subplots(3, 1, figsize=(30, 30),␣
 ↪subplot_kw=dict(projection='3d'))

for i in range(len(axs)):
    ax = axs.ravel()[i]
    np.random.seed(i)
    X = np.random.randn(16, 3)
    X /= np.linalg.norm(X, axis=1).reshape(-1, 1)
    X_final = PGD(X, 0.1, 0.7, 1e-9)

    print('The coordinates when N = 16: ')
    for i in range(n):
        print(X_final[i])
    print('\n')

    potential = U_E(X_final)

    phi, theta = np.mgrid[0.0:2 * np.pi:200j, 0.0:2.0 * np.pi: 200j]
    X = np.sin(phi) * np.cos(theta)
    Y = np.sin(phi) * np.sin(theta)
    Z = np.cos(phi)

    ax.plot_surface(X, Y, Z, rstride=1, cstride=1, alpha=0.1, color='blue',␣
 ↪shade=0)
    ax.scatter(X_final[:, 0], X_final[:, 1], X_final[:, 2], c='r')
    ax.set_box_aspect([1, 1, 1])
    ax.set_title(f"The electrostatic Potential Energy when N=16: U = {potential:
 ↪.6f}")

plt.show()
```

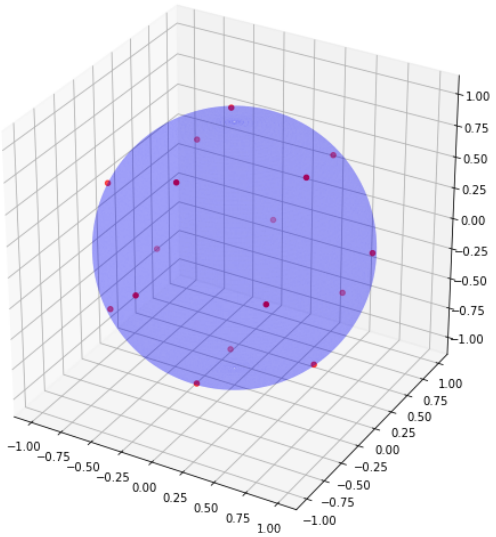
```
The coordinates when N = 16:
[ 0.25354912  0.93184931 -0.25955676]
[ 0.77130609 -0.48913011 -0.40723292]
[ 0.9260882   0.31832217 -0.20256266]
[ 0.09011108 -0.11766563  0.98895642]
[ 0.19119169 -0.97642556  0.10019416]
[-0.11522015  0.81593216  0.56655011]
[ 0.04450567 -0.54165223 -0.83942368]
[ 0.32494889  0.32479362 -0.88821018]
[ 0.74852421 -0.43231333  0.5028088 ]
[-0.70116851  0.21454732  0.67995012]
```

```
[-0.53468481  0.16262061 -0.82925671]
[-0.99650558 -0.04440111 -0.07074727]
[0.65854589 0.47132655 0.58665884]
[-0.58785484 -0.74562546 -0.31379827]


The coordinates when N = 16:
[ 0.75613398 -0.19991363 -0.62313397]
[-0.13320607  0.17364652 -0.97575767]
[0.9614683  0.25200092 0.10988285]
[-0.20985301  0.93390571 -0.28945091]
[-0.10193842 -0.60930546  0.78635578]
[-0.8456172   0.33152862 -0.41835432]
[ 0.52445179  0.65122706 -0.54850127]
[ 0.61935762 -0.09609312  0.77920617]
[ 0.66983541 -0.73279947  0.11968903]
[-0.86956097 -0.27264589  0.41173771]
[ 0.09001109 -0.6814215  -0.72633514]
[-0.2130045  -0.97567952  0.05175488]
[-0.67194265  0.62388176  0.39907971]
[-0.73514155 -0.48777916 -0.47078487]


The coordinates when N = 16:
[ 0.12015824 -0.14737603 -0.98175471]
[ 0.85735684  0.11521736 -0.50166144]
[ 0.39462543 -0.80055867 -0.45097293]
[-0.01682863  0.62628574  0.77941194]
[-0.07040132 -0.94208838  0.32789195]
[-0.47423354 -0.16533383  0.86473538]
[-0.77012806  0.2007013  -0.60549298]
[ 0.01749515  0.64519388 -0.76381855]
[-0.47138368 -0.66696797 -0.5770192 ]
[-0.85369176 -0.50186618  0.13907088]
[ 0.52182981  0.85050672 -0.06581775]
[0.80312931 0.31855421 0.50349432]
[ 0.84586381 -0.50322324  0.17686374]
[ 0.33693476 -0.3325641   0.88083829]
```

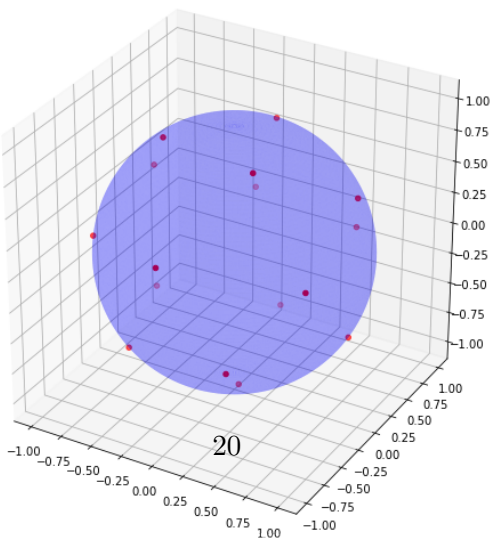
The electrostatic Potential Energy when N=16: U = 92.920354



The electrostatic Potential Energy when N=16: U = 92.911655



The electrostatic Potential Energy when N=16: U = 92.911655

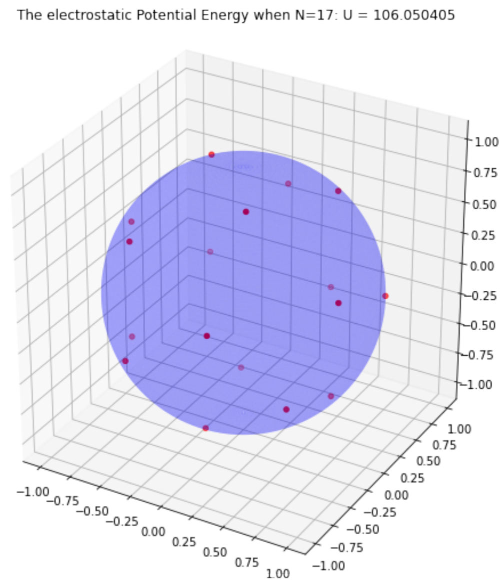


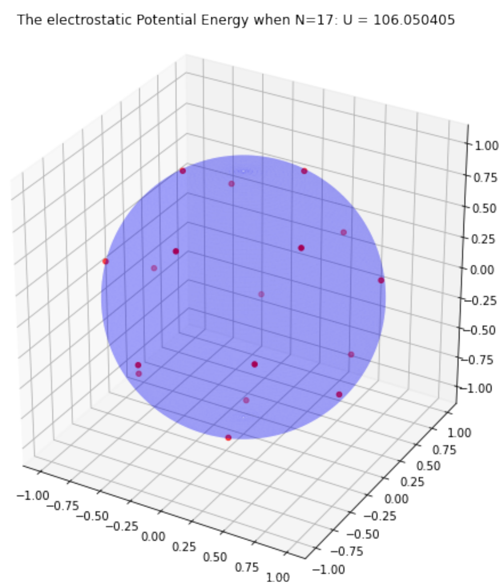
11 N = 17

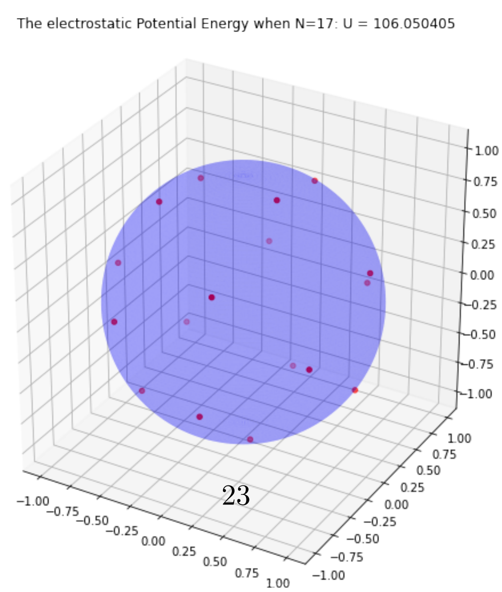
[11]:
```
fig, axs = plt.subplots(3, 1, figsize=(30, 30),␣
 ↪subplot_kw=dict(projection='3d'))

for i in range(len(axs)):
    ax = axs.ravel()[i]
    np.random.seed(i)
    X = np.random.randn(17, 3)
    X /= np.linalg.norm(X, axis=1).reshape(-1, 1)
    X_final = PGD(X, 0.1, 0.7, 1e-9)

    print('The coordinates when N = 17: ')
    for i in range(n):
        print(X_final[i])
    print('\n')

    potential = U_E(X_final)

    phi, theta = np.mgrid[0.0:2 * np.pi:200j, 0.0:2.0 * np.pi: 200j]
    X = np.sin(phi) * np.cos(theta)
    Y = np.sin(phi) * np.sin(theta)
    Z = np.cos(phi)

    ax.plot_surface(X, Y, Z, rstride=1, cstride=1, alpha=0.1, color='blue',␣
 ↪shade=0)
    ax.scatter(X_final[:, 0], X_final[:, 1], X_final[:, 2], c='r')
    ax.set_box_aspect([1, 1, 1])
    ax.set_title(f"The electrostatic Potential Energy when N=17: U = {potential:
 ↪.6f}")

plt.show()
```

```
The coordinates when N = 17:
[ 0.20912925  0.91365782 -0.34856039]
[ 0.68045585 -0.58977138 -0.43491327]
[ 0.86896184  0.49436343 -0.02258595]
[ 0.24006911 -0.38093034  0.89289355]
[ 0.26016772 -0.94236214  0.21039572]
[-0.14356104  0.89557555  0.42111122]
[-0.03258806 -0.47750059 -0.87802688]
[ 0.61240981  0.18585707 -0.76838231]
[ 0.90524555 -0.27913144  0.32033754]
[-0.39327927  0.23526754  0.88880853]
```

```
[-0.24017049  0.38163332 -0.89256605]
[-0.95752752  0.05820242  0.28240666]
[0.50174663 0.44901604 0.73934763]
[-0.44819614 -0.85929239 -0.24644838]


The coordinates when N = 17:
[ 0.88257736 -0.20137245 -0.42486038]
[-0.08068515  0.1804491  -0.98026937]
[ 0.62648925 -0.3049459   0.71729995]
[-0.32042579  0.81830057 -0.47719126]
[-0.11157435 -0.71933017  0.68564952]
[-0.93227981  0.31292473 -0.18147305]
[ 0.56854169  0.56622839 -0.59677949]
[0.9147312  0.32870448 0.23498977]
[ 0.56330205 -0.82463781  0.05160705]
[-0.8754097  -0.4165583   0.24522855]
[ 0.21722917 -0.5867311  -0.78010134]
[-0.28834638 -0.94569756 -0.1500416 ]
[-0.49561325  0.01293178  0.86844705]
[-0.67185494 -0.33504172 -0.66057398]


The coordinates when N = 17:
[ 0.1951258  -0.24202502 -0.95044716]
[ 0.83828077  0.11719719 -0.5324943 ]
[ 0.16993549 -0.89524644 -0.41189287]
[0.1793193  0.69789828 0.69338487]
[ 0.23094032 -0.82536986  0.51520011]
[-0.48872969 -0.31409653  0.81393283]
[-0.71290923  0.40218374 -0.57446381]
[ 0.08065342  0.58486741 -0.80710912]
[-0.58791851 -0.42791957 -0.6864668 ]
[-0.54844563 -0.8302875   0.09914667]
[ 0.54798628  0.83064369 -0.09870204]
[0.89954672 0.18746934 0.39455157]
[ 0.83375804 -0.55197044 -0.01327277]
[ 0.3205669  -0.09940642  0.94199534]
```


The electrostatic Potential Energy when N=17: U = 106.050405
−1.00 −0.75 −0.50 −0.25 0.00 0.25 0.50 0.75 1.00
−1.00 −0.75 −0.50 −0.25 0.00 0.25 0.50 0.75 1.00
1.00 0.75 0.50 0.25 0.00 −0.25 −0.50 −0.75 −1.00


The electrostatic Potential Energy when N=17: U = 106.050405
−1.00 −0.75 −0.50 −0.25 0.00 0.25 0.50 0.75 1.00
−1.00 −0.75 −0.50 −0.25 0.00 0.25 0.50 0.75 1.00
1.00 0.75 0.50 0.25 0.00 −0.25 −0.50 −0.75 −1.00

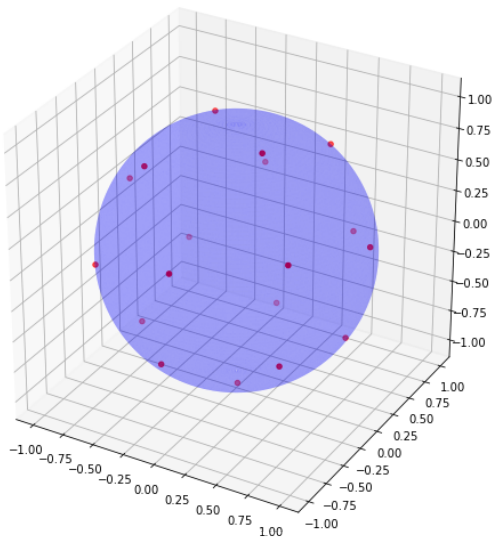

The electrostatic Potential Energy when N=17: U = 106.050405
−1.00 −0.75 −0.50 −0.25 0.00 0.25 0.50 0.75 1.00
−1.00 −0.75 −0.50 −0.25 0.00 0.25 0.50 0.75 1.00
1.00 0.75 0.50 0.25 0.00 −0.25 −0.50 −0.75 −1.00

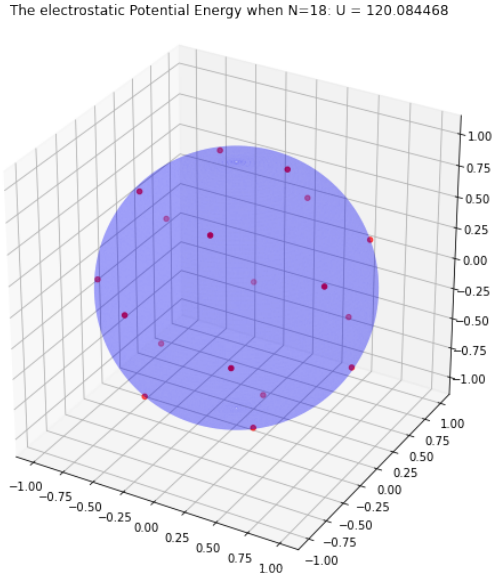
12 N = 18

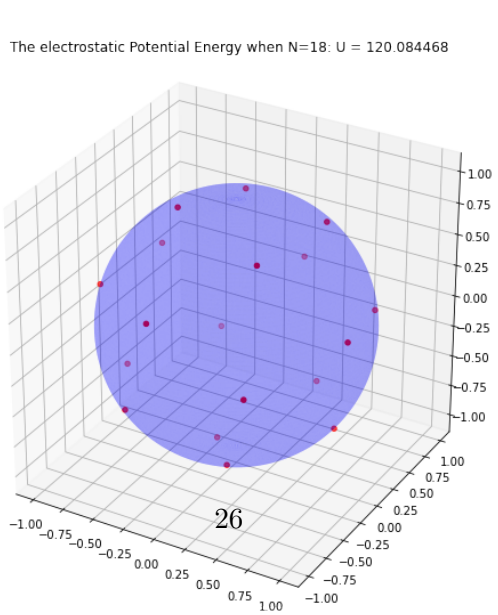
[12]:
```
fig, axs = plt.subplots(3, 1, figsize=(30, 30),␣
 ↪subplot_kw=dict(projection='3d'))

for i in range(len(axs)):
    ax = axs.ravel()[i]
    np.random.seed(i)
    X = np.random.randn(18, 3)
    X /= np.linalg.norm(X, axis=1).reshape(-1, 1)
    X_final = PGD(X, 0.1, 0.7, 1e-9)

    print('The coordinates when N = 18: ')
    for i in range(n):
        print(X_final[i])
    print('\n')

    potential = U_E(X_final)

    phi, theta = np.mgrid[0.0:2 * np.pi:200j, 0.0:2.0 * np.pi: 200j]
    X = np.sin(phi) * np.cos(theta)
    Y = np.sin(phi) * np.sin(theta)
    Z = np.cos(phi)

    ax.plot_surface(X, Y, Z, rstride=1, cstride=1, alpha=0.1, color='blue',␣
 ↪shade=0)
    ax.scatter(X_final[:, 0], X_final[:, 1], X_final[:, 2], c='r')
    ax.set_box_aspect([1, 1, 1])
    ax.set_title(f"The electrostatic Potential Energy when N=18: U = {potential:
 ↪.6f}")

plt.show()
```

```
The coordinates when N = 18:
[ 0.47144316  0.87102148 -0.13806857]
[ 0.10360632 -0.16419261 -0.98097223]
[0.95986125 0.18551854 0.21035505]
[ 0.30264708 -0.17314905  0.93724285]
[ 0.0259317  -0.95293222  0.30207241]
[-0.29671408  0.93524961  0.19305162]
[-0.68884151 -0.16742072 -0.70531389]
[ 0.76512037  0.22058445 -0.60492422]
[ 0.73006125 -0.56821723  0.37965741]
[-0.40076295  0.39518728  0.82656886]
```

```
[-0.02808174  0.63713689 -0.77023893]
[-0.95745675  0.13036512  0.25745194]
[0.40852285 0.61617808 0.67337483]
[-0.11169655 -0.84753132 -0.51885889]


The coordinates when N = 18:
[ 0.92833376 -0.00275138 -0.37173763]
[ 0.10191696  0.2099771  -0.97237984]
[ 0.8611922  -0.30110234  0.40949405]
[-0.32898464  0.82597612 -0.45774726]
[ 0.15690946 -0.62357962  0.76585109]
[-0.68956061  0.09823719 -0.7175344 ]
[ 0.51770835  0.72398314 -0.45587989]
[0.76769636 0.54361482 0.3393011 ]
[ 0.46807266 -0.88224064 -0.05059086]
[-0.61048589 -0.28542023  0.73881139]
[-0.35780999 -0.66953214 -0.65092144]
[-0.93946853 -0.34122833 -0.03101776]
[-0.42210535  0.49815838  0.75740696]
[ 0.42250509 -0.49762543 -0.75753441]


The coordinates when N = 18:
[-0.16390975  0.00578542 -0.98645837]
[ 0.80314691 -0.02102031 -0.5954101 ]
[ 0.27996204 -0.61809306 -0.73456261]
[-0.27941603  0.61847978  0.73444499]
[-0.18876078 -0.89484058  0.4045117 ]
[-0.37390248 -0.1618435   0.91323798]
[-0.85055651 -0.10044216 -0.51620247]
[ 0.30230321  0.64329782 -0.70340649]
[-0.42519071 -0.81367238 -0.39642164]
[ 0.41096691 -0.43614367  0.80055287]
[0.75830873 0.65166783 0.01723074]
[0.49494281 0.39812951 0.77234999]
[ 0.54133415 -0.84079075 -0.00531454]
[ 0.96425016 -0.1513131   0.21754536]
```


The electrostatic Potential Energy when N=18: U = 120.084468


The electrostatic Potential Energy when N=18: U = 120.084468

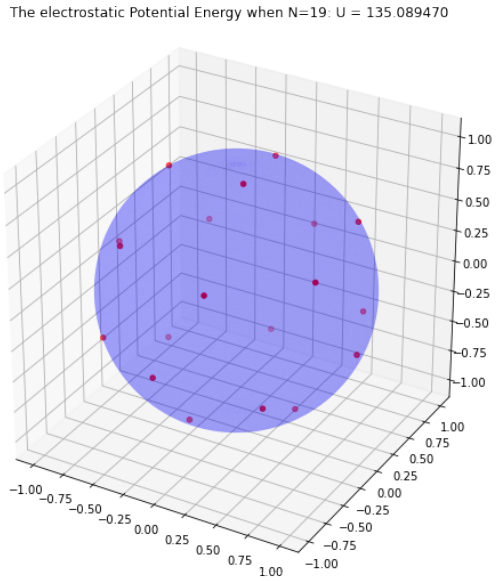

The electrostatic Potential Energy when N=18: U = 120.084468

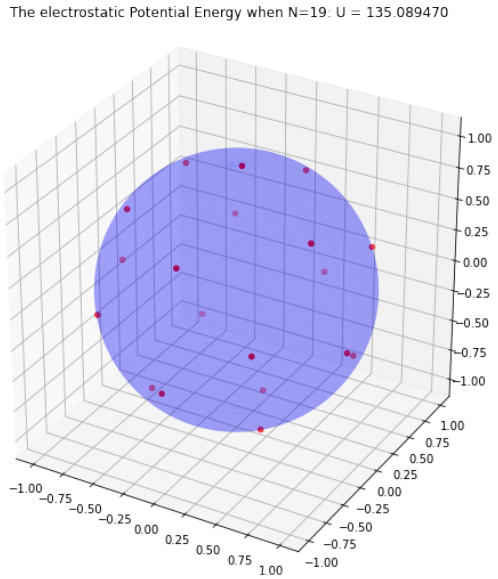
13 N = 19

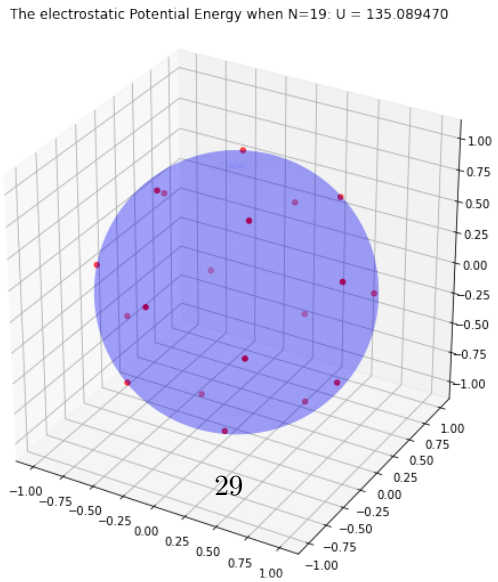
[13]:
```
fig, axs = plt.subplots(3, 1, figsize=(30, 30),␣
 ↪subplot_kw=dict(projection='3d'))

for i in range(len(axs)):
    ax = axs.ravel()[i]
    np.random.seed(i)
    X = np.random.randn(19, 3)
    X /= np.linalg.norm(X, axis=1).reshape(-1, 1)
    X_final = PGD(X, 0.1, 0.7, 1e-9)

    print('The coordinates when N = 19: ')
    for i in range(n):
        print(X_final[i])
    print('\n')

    potential = U_E(X_final)

    phi, theta = np.mgrid[0.0:2 * np.pi:200j, 0.0:2.0 * np.pi: 200j]
    X = np.sin(phi) * np.cos(theta)
    Y = np.sin(phi) * np.sin(theta)
    Z = np.cos(phi)

    ax.plot_surface(X, Y, Z, rstride=1, cstride=1, alpha=0.1, color='blue',␣
 ↪shade=0)
    ax.scatter(X_final[:, 0], X_final[:, 1], X_final[:, 2], c='r')
    ax.set_box_aspect([1, 1, 1])
    ax.set_title(f"The electrostatic Potential Energy when N=19: U = {potential:
 ↪.6f}")

plt.show()
```

```
The coordinates when N = 19:
[-0.1089918   0.69834254 -0.70741676]
[ 0.43464711  0.08556325 -0.89652709]
[ 0.97411337 -0.0188114  -0.225276  ]
[ 0.15635544 -0.17638551  0.97182361]
[ 0.24251556 -0.85526379  0.45794111]
[-0.04431542  0.63167302  0.77396727]
[-0.69056934  0.20314415 -0.6941516 ]
[ 0.65635088  0.68111755 -0.32447251]
[ 0.81433838 -0.34257889  0.46850049]
[-0.61324901  0.11572877  0.78136579]
```

```
[-0.14984002 -0.40635282 -0.90134641]
[-0.99390877  0.0411427   0.10223817]
[0.71214215 0.46525973 0.52572515]
[-0.10458909 -0.96538417 -0.23894462]


The coordinates when N = 19:
[ 0.97597648 -0.16915907 -0.13731391]
[ 0.06504385  0.27276311 -0.95987998]
[ 0.70698453 -0.214062    0.67405515]
[-0.58599899  0.51076427 -0.6290668 ]
[ 0.00421766 -0.81676812  0.57695065]
[-0.99416886  0.07823945 -0.0742082 ]
[ 0.71375039  0.43049832 -0.55247767]
[0.78616438 0.53732335 0.30533455]
[ 0.58538443 -0.80896491  0.05385947]
[-0.6492503  -0.38936027  0.6533549 ]
[-0.50945943 -0.32493235 -0.79678734]
[-0.72026314 -0.69066085 -0.06487372]
[-0.63683426  0.3854634   0.66772756]
[ 0.44375468 -0.42929267 -0.7866318 ]


The coordinates when N = 19:
[-0.30392618  0.02249824 -0.9524299 ]
[ 0.91161085 -0.18630501 -0.36640976]
[ 0.27767846 -0.64298151 -0.71377129]
[-0.19069086  0.42281578  0.8859254 ]
[-0.16791588 -0.93782998  0.3037749 ]
[-0.44910887 -0.31370564  0.83659428]
[-0.89568191 -0.02814478 -0.44380376]
[ 0.45759892  0.19143115 -0.86830717]
[-0.43726368 -0.76455951 -0.4735496 ]
[ 0.3302753  -0.40198085  0.85400798]
[ 0.10444542  0.81929942 -0.56377266]
[0.53747495 0.52175628 0.66248854]
[ 0.54827404 -0.83422255  0.05889239]
[ 0.90618142 -0.12307016  0.40458493]
```


The electrostatic Potential Energy when N=19: U = 135.089470


The electrostatic Potential Energy when N=19: U = 135.089470


The electrostatic Potential Energy when N=19: U = 135.089470

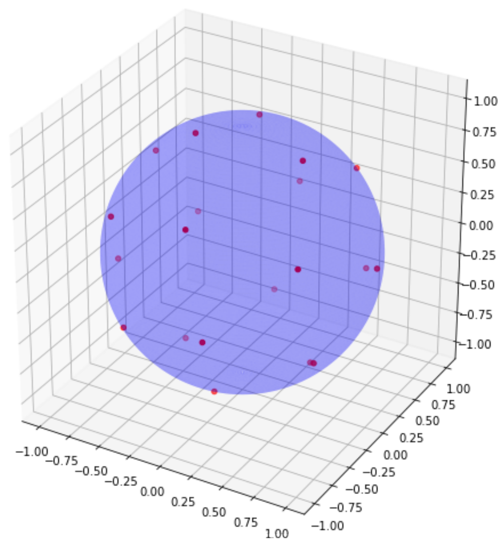
14 N = 20

[14]:
```
fig, axs = plt.subplots(3, 1, figsize=(30, 30),␣
 ↪subplot_kw=dict(projection='3d'))

for i in range(len(axs)):
    ax = axs.ravel()[i]
    np.random.seed(i)
    X = np.random.randn(20, 3)
    X /= np.linalg.norm(X, axis=1).reshape(-1, 1)
    X_final = PGD(X, 0.1, 0.7, 1e-9)

    print('The coordinates when N = 20: ')
    for i in range(n):
        print(X_final[i])
    print('\n')

    potential = U_E(X_final)

    phi, theta = np.mgrid[0.0:2 * np.pi:200j, 0.0:2.0 * np.pi: 200j]
    X = np.sin(phi) * np.cos(theta)
    Y = np.sin(phi) * np.sin(theta)
    Z = np.cos(phi)

    ax.plot_surface(X, Y, Z, rstride=1, cstride=1, alpha=0.1, color='blue',␣
 ↪shade=0)
    ax.scatter(X_final[:, 0], X_final[:, 1], X_final[:, 2], c='r')
    ax.set_box_aspect([1, 1, 1])
    ax.set_title(f"The electrostatic Potential Energy when N=20: U = {potential:
 ↪.6f}")

plt.show()
```

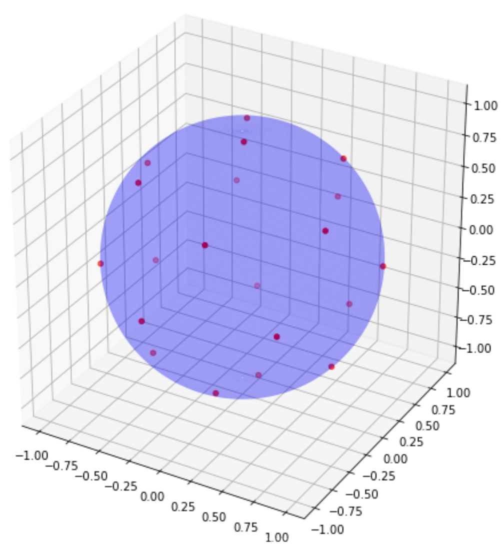
```
The coordinates when N = 20:
[-0.13303418  0.69940098 -0.7022394 ]
[ 0.45555907  0.18207526 -0.87138656]
[0.98475378 0.15737405 0.07411745]
[ 0.48829756 -0.02278507  0.8723797 ]
[ 0.02521686 -0.80958485  0.58646098]
[-0.215428    0.61521135  0.75835729]
[-0.48825139  0.02279729 -0.87240522]
[ 0.60690456  0.72720613 -0.32068379]
[ 0.75055669 -0.55386542  0.36041331]
[-0.27884598 -0.16131176  0.94669078]
```

```
[ 0.07329775 -0.52225302 -0.84963476]
[-0.8464309   0.23101246  0.47977908]
[0.54422764 0.66778844 0.50781382]
[-0.50445185 -0.78486151 -0.35988991]


The coordinates when N = 20:
[ 0.79991628 -0.14146622 -0.58319915]
[ 0.09207421  0.07216537 -0.99313368]
[ 0.79155194 -0.24843873  0.55832224]
[-0.27372002  0.6918106  -0.66818818]
[ 0.18013738 -0.81433043  0.55173949]
[-0.87723199  0.25507508 -0.40669489]
[ 0.51948079  0.64304425 -0.56270223]
[0.9451527  0.32449835 0.03724501]
[ 0.68664262 -0.72426639 -0.06292943]
[-0.54246036 -0.49230708  0.68071323]
[-0.53199587 -0.35600387 -0.76827185]
[-0.86492538 -0.49907604 -0.05317132]
[-0.88304594  0.17611355  0.43498722]
[ 0.18894285 -0.69793452 -0.69078796]


The coordinates when N = 20:
[-0.22549122 -0.23863558 -0.94456697]
[ 0.95129489  0.01415034 -0.30795746]
[ 0.51550279 -0.53746628 -0.66737311]
[0.11188478 0.81060537 0.57480495]
[-0.45463185 -0.83171594  0.3186824 ]
[-0.3808422  -0.09082506  0.92016848]
[-0.81686992 -0.37388879 -0.43923879]
[-0.32371072  0.52833237 -0.78490527]
[-0.19995086 -0.88527474 -0.4198908 ]
[ 0.17365737 -0.64296137  0.74595161]
[ 0.45462232  0.8317395  -0.31863449]
[0.37170048 0.14795826 0.91648628]
[ 0.45812734 -0.8880861   0.03771488]
[ 0.85901714 -0.31489519  0.4036466 ]
```

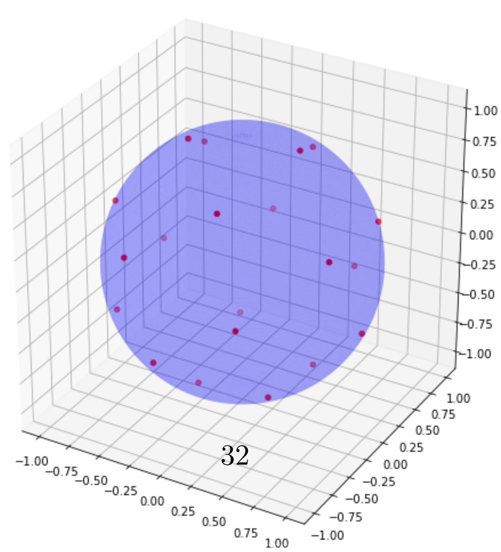
The electrostatic Potential Energy when N=20: U = 150.881568



The electrostatic Potential Energy when N=20: U = 150.881568



The electrostatic Potential Energy when N=20: U = 150.881568



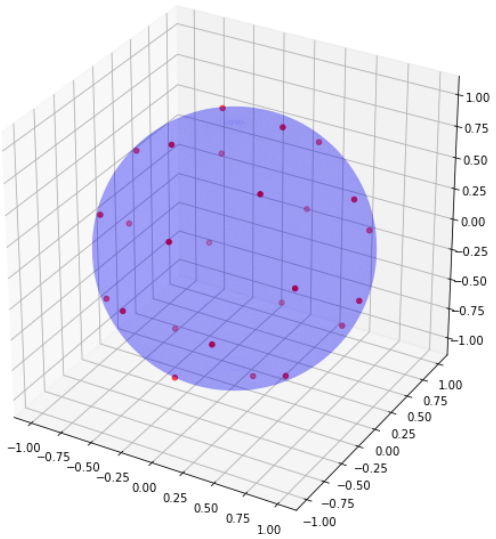
15 N = 25

[15]:
```
fig, axs = plt.subplots(3, 1, figsize=(30, 30),␣
 ↪subplot_kw=dict(projection='3d'))

for i in range(len(axs)):
    ax = axs.ravel()[i]
    np.random.seed(i)
    X = np.random.randn(25, 3)
    X /= np.linalg.norm(X, axis=1).reshape(-1, 1)
    X_final = PGD(X, 0.1, 0.7, 1e-9)

    print('The coordinates when N = 25: ')
    for i in range(n):
        print(X_final[i])
    print('\n')

    potential = U_E(X_final)

    phi, theta = np.mgrid[0.0:2 * np.pi:200j, 0.0:2.0 * np.pi: 200j]
    X = np.sin(phi) * np.cos(theta)
    Y = np.sin(phi) * np.sin(theta)
    Z = np.cos(phi)

    ax.plot_surface(X, Y, Z, rstride=1, cstride=1, alpha=0.1, color='blue',␣
 ↪shade=0)
    ax.scatter(X_final[:, 0], X_final[:, 1], X_final[:, 2], c='r')
    ax.set_box_aspect([1, 1, 1])
    ax.set_title(f"The electrostatic Potential Energy when N=25: U = {potential:
 ↪.6f}")

plt.show()
```

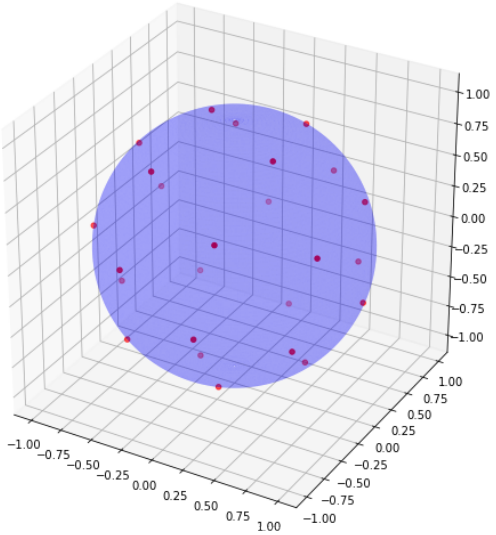
```
The coordinates when N = 25:
[-0.56398697  0.7896451   0.2416181 ]
[ 0.17958413 -0.04430656 -0.98274436]
[ 0.99034439  0.01111983 -0.13818226]
[0.26837941 0.20187139 0.94192379]
[-0.00512425 -0.8736354   0.48655414]
[ 0.66631122  0.38554589 -0.63826619]
[-0.52519605  0.0483102  -0.84960887]
[ 0.693827    0.72005418 -0.01122847]
[ 0.79926621 -0.56570214  0.20286598]
[-0.35670888  0.45046948  0.8184351 ]
```

```
[ 0.03080528  0.63884321 -0.76871996]
[-0.7452131  -0.57257539  0.34178188]
[0.22348829 0.81682837 0.53183118]
[-0.35077839 -0.92666476 -0.13508128]


The coordinates when N = 25:
[ 0.76730917 -0.53914591 -0.34721509]
[-0.25604677 -0.04515062 -0.96560938]
[ 0.86309193 -0.37512878  0.33815784]
[-0.25887878  0.95017596 -0.17363014]
[0.28060357 0.51963111 0.80699761]
[-0.86815438 -0.11958646 -0.4816711 ]
[ 0.60840605  0.72695817 -0.31839267]
[ 0.96034453  0.13001823 -0.24664479]
[ 0.53837665  0.07277227 -0.8395563 ]
[-0.81544616  0.06427139  0.57525364]
[ 0.19410574 -0.55822619 -0.80666379]
[-0.37879844 -0.91229794  0.15564131]
[-0.33665977 -0.55116343  0.76346517]
[ 0.22969287 -0.9540104  -0.19262747]


The coordinates when N = 25:
[ 0.3427514  -0.21226972 -0.91513007]
[ 0.88569275 -0.21326828 -0.41238937]
[-0.12596979 -0.71544632 -0.68721771]
[-0.59698234  0.08298761  0.79795059]
[ 0.04773941 -0.3369842   0.9402992 ]
[0.05822952 0.36943899 0.92742879]
[-0.66144638 -0.71729668 -0.21903002]
[ 0.41202215  0.88414097 -0.22030092]
[-0.37143054 -0.08213182 -0.92482091]
[ 0.51199954 -0.79727522 -0.31970094]
[-0.63462452  0.54545834 -0.54747322]
[-0.35428172  0.77568124  0.52230553]
[ 0.88522954 -0.38929663  0.25459143]
[ 0.38399284 -0.76637625  0.51499218]
```

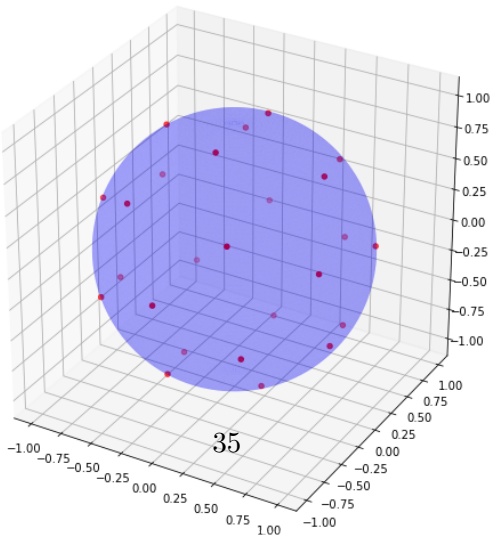
The electrostatic Potential Energy when N=25: U = 243.812761



The electrostatic Potential Energy when N=25: U = 243.812761



The electrostatic Potential Energy when N=25: U = 243.812762

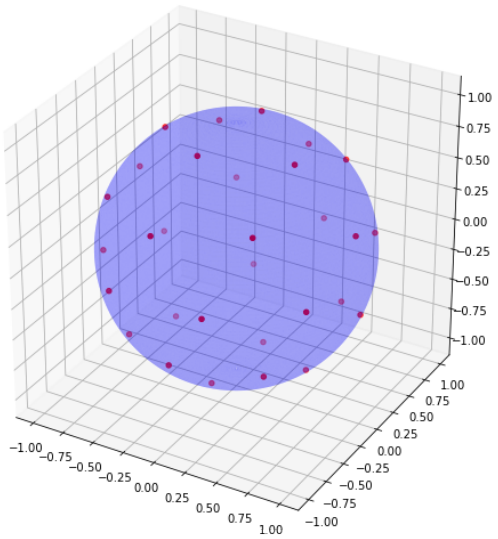


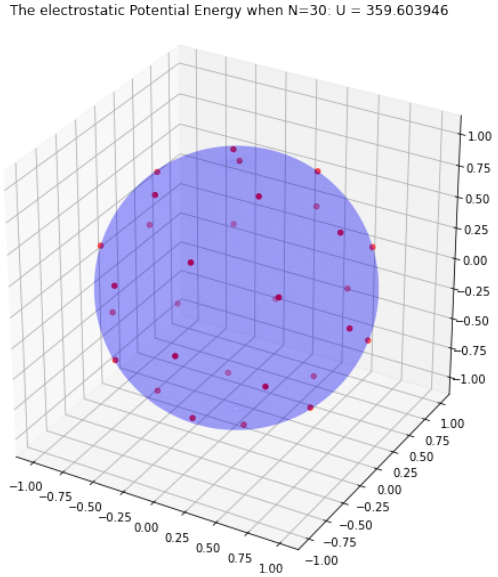
16 N = 30

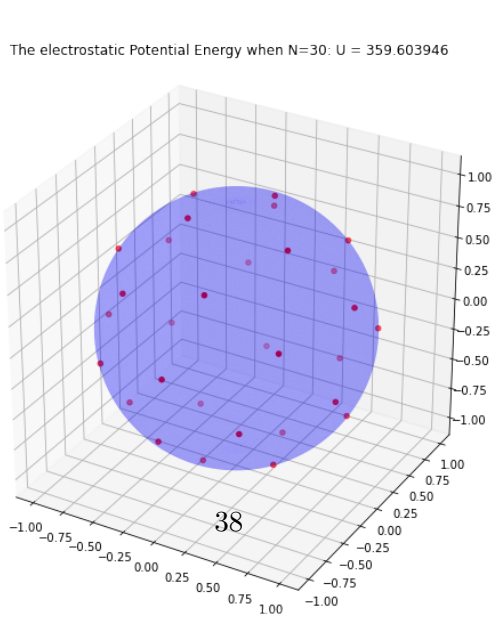
[16]:
```
fig, axs = plt.subplots(3, 1, figsize=(30, 30),␣
 ↪subplot_kw=dict(projection='3d'))

for i in range(len(axs)):
    ax = axs.ravel()[i]
    np.random.seed(i)
    X = np.random.randn(30, 3)
    X /= np.linalg.norm(X, axis=1).reshape(-1, 1)
    X_final = PGD(X, 0.1, 0.7, 1e-9)

    print('The coordinates when N = 30: ')
    for i in range(n):
        print(X_final[i])
    print('\n')

    potential = U_E(X_final)

    phi, theta = np.mgrid[0.0:2 * np.pi:200j, 0.0:2.0 * np.pi: 200j]
    X = np.sin(phi) * np.cos(theta)
    Y = np.sin(phi) * np.sin(theta)
    Z = np.cos(phi)

    ax.plot_surface(X, Y, Z, rstride=1, cstride=1, alpha=0.1, color='blue',␣
 ↪shade=0)
    ax.scatter(X_final[:, 0], X_final[:, 1], X_final[:, 2], c='r')
    ax.set_box_aspect([1, 1, 1])
    ax.set_title(f"The electrostatic Potential Energy when N=30: U = {potential:
 ↪.6f}")

plt.show()
```

```
The coordinates when N = 30:
[-0.49730475  0.61622938  0.61069578]
[ 0.58554008 -0.03021837 -0.81008003]
[0.93350719 0.01703001 0.35815401]
[ 0.51248822 -0.09584694  0.8533283 ]
[-0.17445267 -0.86140089  0.47702702]
[ 0.95013157  0.08167575 -0.30096356]
[-0.00977157 -0.33490255 -0.9422021 ]
[ 0.17756517  0.96807794 -0.17690595]
[ 0.85035266 -0.52611271  0.01028428]
[0.00977158 0.33490255 0.9422021 ]
```

```
[ 0.03354385  0.33152181 -0.94285105]
[-0.56281098 -0.01795999  0.82639049]
[0.08501395 0.89055192 0.44686676]
[-0.04872884 -0.85196099 -0.52133287]


The coordinates when N = 30:
[ 0.73720967 -0.24647676 -0.62910342]
[-0.46556097 -0.32242027 -0.82419546]
[ 0.98644428 -0.15845564  0.04265539]
[-0.14343356  0.82401654 -0.54810907]
[ 0.64806568 -0.72784187 -0.22418092]
[-0.90913041 -0.20287833 -0.36376131]
[ 0.87744664  0.33588331 -0.34244678]
[0.75635167 0.6055554  0.24745667]
[ 0.09578562 -0.99536399 -0.00869685]
[-0.90316739 -0.34502971  0.2554274 ]
[ 0.09306846 -0.7696715  -0.63162017]
[-0.48901387 -0.82349436  0.28761514]
[-0.73849091  0.15903287  0.6552402 ]
[ 0.19322319 -0.23033819 -0.9537343 ]


The coordinates when N = 30:
[ 0.52760116 -0.39866493 -0.75013552]
[ 0.94502913 -0.27740271 -0.17311173]
[-0.0759419  -0.34384362 -0.93595106]
[0.12481538 0.31741469 0.94003672]
[0.55676791 0.60349492 0.57079188]
[ 0.6976074  -0.63967279  0.32274236]
[-0.48116171 -0.72272727  0.49613376]
[ 0.43254142  0.74944417 -0.50123982]
[-0.68688036 -0.3387602  -0.64299059]
[ 0.49593458 -0.82262196 -0.2781043 ]
[-0.1934109   0.77877692 -0.59674009]
[-0.67083892 -0.73324065 -0.1110554 ]
[ 0.16972737 -0.72505336  0.66745055]
[0.25933556 0.96307323 0.07235341]
```


The electrostatic Potential Energy when N=30: U = 359.603946


The electrostatic Potential Energy when N=30: U = 359.603946


The electrostatic Potential Energy when N=30: U = 359.603946

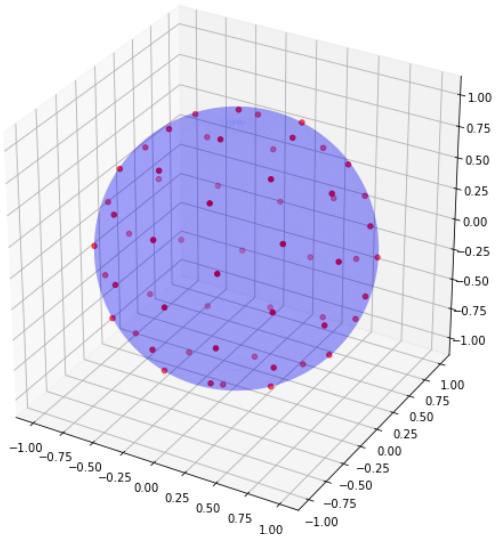
17 N = 60

[17]:
```
fig, axs = plt.subplots(3, 1, figsize=(30, 30),␣
 ↪subplot_kw=dict(projection='3d'))

for i in range(len(axs)):
    ax = axs.ravel()[i]
    np.random.seed(i)
    X = np.random.randn(60, 3)
    X /= np.linalg.norm(X, axis=1).reshape(-1, 1)
    X_final = PGD(X, 0.1, 0.7, 1e-9)

    print('The coordinates when N = 60: ')
    for i in range(n):
        print(X_final[i])
    print('\n')

    potential = U_E(X_final)

    phi, theta = np.mgrid[0.0:2 * np.pi:200j, 0.0:2.0 * np.pi: 200j]
    X = np.sin(phi) * np.cos(theta)
    Y = np.sin(phi) * np.sin(theta)
    Z = np.cos(phi)

    ax.plot_surface(X, Y, Z, rstride=1, cstride=1, alpha=0.1, color='blue',␣
 ↪shade=0)
    ax.scatter(X_final[:, 0], X_final[:, 1], X_final[:, 2], c='r')
    ax.set_box_aspect([1, 1, 1])
    ax.set_title(f"The electrostatic Potential Energy when N=60: U = {potential:
 ↪.6f}")

plt.show()
```

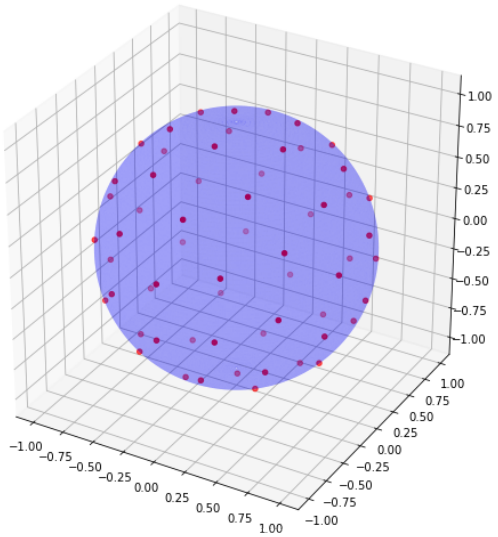
```
The coordinates when N = 60:
[ 0.12165134  0.87188475 -0.47436055]
[-0.35327811 -0.07190707 -0.93275074]
[0.36981351 0.13074252 0.91986105]
[-0.30218115 -0.53142105  0.79137742]
[ 0.91293836 -0.37740994 -0.15525876]
[0.62240776 0.74535101 0.23887331]
[-0.39141589  0.76481669 -0.51171187]
[ 0.69505963 -0.7163584   0.06101442]
[ 0.67397066 -0.54288704  0.50103614]
[ 0.48105134 -0.35113837 -0.80330035]
```

```
[-0.23688163 -0.6043526  -0.76068721]
[ 0.01380204 -0.99901885  0.04208127]
[-0.18936434  0.97153425 -0.14234586]
[-0.14388948 -0.87489307  0.46244777]


The coordinates when N = 60:
[-0.73040561 -0.42310722  0.53617901]
[ 0.72514557 -0.68849388 -0.01183586]
[ 0.97461182  0.10782814 -0.19622665]
[-0.48855993  0.73917465  0.46360547]
[-0.4602064  -0.12340782  0.87919314]
[-0.15598346 -0.44945638 -0.87957838]
[ 0.04528046 -0.06536292 -0.99683367]
[ 0.69802934 -0.56352012  0.44181458]
[ 0.38684219 -0.51484546  0.7650407 ]
[-0.35760124 -0.73595387 -0.57488542]
[ 0.75099374 -0.14235704 -0.64478126]
[0.94127213 0.20527221 0.26808598]
[-0.79711018  0.03605951  0.60275623]
[-0.72025276  0.69364307  0.00976019]


The coordinates when N = 60:
[-0.11712685  0.51305204 -0.85032871]
[ 0.674332   -0.39372062 -0.62470828]
[-0.11688488 -0.99294295 -0.02005529]
[-0.23672171  0.24492235  0.94019991]
[ 0.98088438  0.18742386 -0.05232714]
[ 0.3735046  -0.91453365  0.15531424]
[-0.61178127 -0.17499767 -0.77142692]
[ 0.29668326 -0.66856531 -0.6819087 ]
[-0.12201347 -0.47171179 -0.87327012]
[-0.98937337 -0.14501814  0.01049125]
[0.09713834 0.87939508 0.46607771]
[0.50494419 0.53709558 0.67569202]
[ 0.00124071 -0.60698592  0.79471162]
[ 0.71495086 -0.3638117   0.59706474]
```

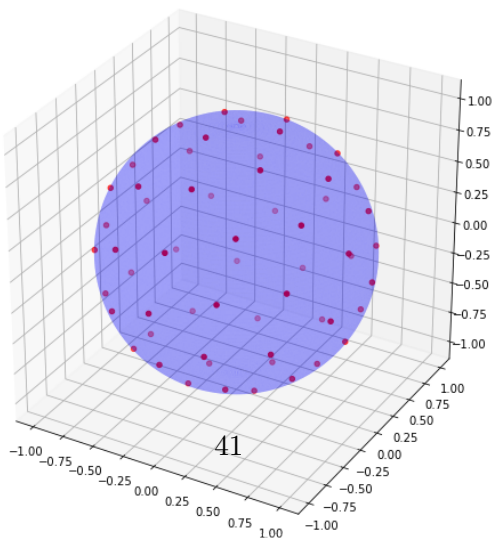
The electrostatic Potential Energy when N=60: U = 1543.835148



The electrostatic Potential Energy when N=60: U = 1543.830411



The electrostatic Potential Energy when N=60: U = 1543.835102

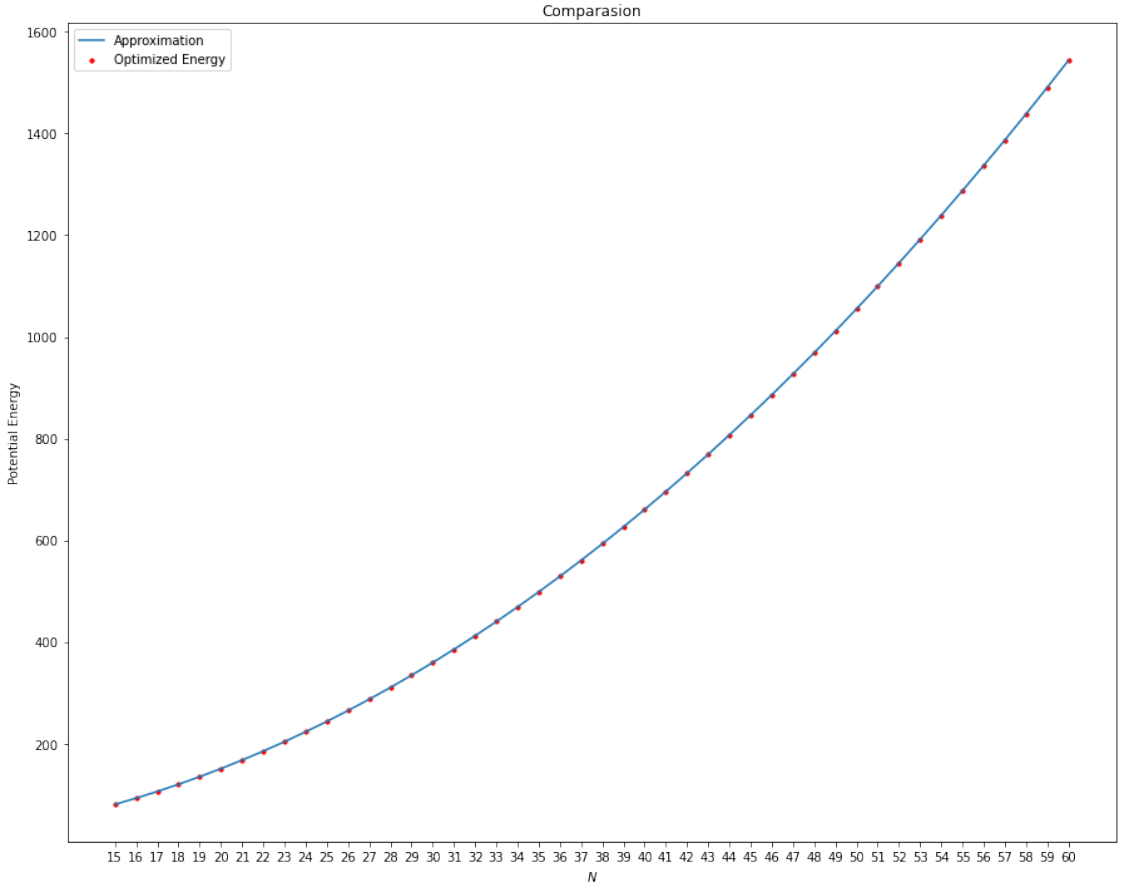


17.1 From above we could see there may be some difference when N is larger than 14 but the difference is comparable.

18 Approximation & Visualizing

[18]:
```
def approx(N):
    a = 1.10461
    b = 0.137
    return (1/2) * (N ** 2) * (1 - a * N ** (-1/2) + b * N ** (-3/2))
```

[19]:
```
energy = []

for i in np.arange(15, 61):
    X = np.random.randn(i, 3)
    X /= np.linalg.norm(X, axis=1).reshape(-1, 1)
    X_final = PGD(X, 0.1, 0.7, 1e-8)

    potential = U_E(X_final)
    energy.append((i, potential))

energy = np.array(energy)

plt.figure(figsize=(15, 12))
plt.plot(np.arange(15, 61), approx(np.arange(15, 61)), label="Approximation")
plt.scatter(energy[:, 0], energy[:, 1], s=10, c='red', label="Optimized Energy")
plt.xticks(np.arange(15, 61))
plt.legend()
plt.xlabel("$N$")
plt.ylabel("Potential Energy")
plt.title("Comparasion")
plt.show()
```


Comparasion
Approximation
Optimized Energy
Potential Energy
N
1600
1400
1200
1000
800
600
400
200
15 16 17 18 19 20 21 22 23 24 25 26 27 28 29 30 31 32 33 34 35 36 37 38 39 40 41 42 43 44 45 46 47 48 49 50 51 52 53 54 55 56 57 58 59 60